# HR Analytics Case Study

## Problem Definition

### Introduction

Every year a lot of companies hire a number of employees. The companies invest time and money in training those employees, not just this but there are training programs within the companies for their existing employees as well. The aim of these programs is to increase the effectiveness of their employees. But where HR Analytics fit in this? and is it just about improving the performance of employees?

### HR Analytics

Human resource analytics (HR analytics) is an area in the field of analytics that refers to applying analytic processes to the human resource department of an organization in the hope of improving employee performance and therefore getting a better return on investment. HR analytics does not just deal with gathering data on employee efficiency. Instead, it aims to provide insight into each process by gathering data and then using it to make relevant decisions about how to improve these processes.

### Attrition in HR

Attrition in human resources refers to the gradual loss of employee’s overtime. In general, relatively high attrition is problematic for companies. HR professionals often assume a leadership role in designing company compensation programs, work culture, and motivation systems that help the organization retain top employees.

How does Attrition affect companies? and how does HR Analytics help in analysing attrition? We will discuss the first question here and for the second question, we will write the code and try to understand the process step by step.

### Attrition affecting Companies

A major problem in high employee attrition is its cost to an organization. Job postings, hiring processes, paperwork, and new hire training are some of the common expenses of losing employees and replacing them. Additionally, regular employee turnover prohibits your organization from increasing its collective knowledge base and experience over time. This is especially concerning if your business is customer-facing, as customers often prefer to interact with familiar people. Errors and issues are more likely if you constantly have new workers.

## Data Analysis

### Size and Shape of Data

The data set contains 1470 record with 35 features.

### Features and What those features are describing

All the features and their meaning explained below –

* **Age**: Age of employee
* **Attrition**: Left or Not
* **BusinessTravel**: Does Require Travel
* **DailyRate**: Daily income
* **Department**: Department in which employee is working
* **DistanceFromHome**: Office to Home distance
* **Education**: Highest qualification of employee
* **EducationField**: Field of Education
* **EmployeeCount**: Total Employee Count
* **EmployeeNumber**: Unique Employee Number
* **EnvironmentSatisfaction**: Employee feedback on Environment Satisfaction
* **Gender**: Gender of employee
* **HourlyRate**: Hourly Income of employee
* **JobInvolvement**: Employees total job involvement.
* **JobLevel**: Level of job profile
* **JobRole**: His role in that profile and department
* **JobSatisfaction**: Employees job satisfaction level
* **MaritalStatus**: Is employee married/Divorced/Single
* **MonthlyIncome**: Monthly income
* **MonthlyRate**: Rate of employee
* **NumCompaniesWorked**: Total number of companies employees has worked
* **Over18**: Is employee over 18
* **OverTime**: Does employee does overtime
* **PercentSalaryHike**: Employees salary hike
* **PerformanceRating**: Total performance rating of employee
* **RelationshipSatisfaction**: Employees relationship satisfaction
* **StandardHours**: Standard working hours that employee should work to get full day salary
* **StockOptionLevel**: Stock option level of employees
* **TotalWorkingYears**: Total working experience of employee.
* **TrainingTimesLastYear**: Training given to employee
* **WorkLifeBalance**: Work life balance rating
* **YearsAtCompany**: Years employee was working in this organisation
* **YearsInCurrentRole**: Years employee was working in current role
* **YearsSinceLastPromotion**: Time passed after last promotion
* **YearsWithCurrManager**: Years with current manager.

## Pre-processing Pipeline This preprocessing step involves below mentioned steps.

### Importing Libraries

# To Read and Process Data

import pandas as pd

import numpy as np

# For data Visualization

import seaborn as sns

import matplotlib.pyplot as plt

# Getting over warning messages

import warnings

warnings.filterwarnings('ignore')

# For Encoding Categorical Data

from sklearn.preprocessing import LabelEncoder

# for scaling

from sklearn.preprocessing import StandardScaler

# To display all columns

pd.pandas.set\_option('display.max\_columns',None)

# For handling outliers

# importing required libraries

from scipy import stats

import statsmodels.api as sm

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

from scipy.stats import zscore

# For machine learning and finding

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

### Reading Data

df = pd.read\_csv('WA\_Fn-UseC\_-HR-Employee-Attrition.csv')

### Data Overview

Overall Data Analysis

# getting to know size of data set, to know overall records, and columns

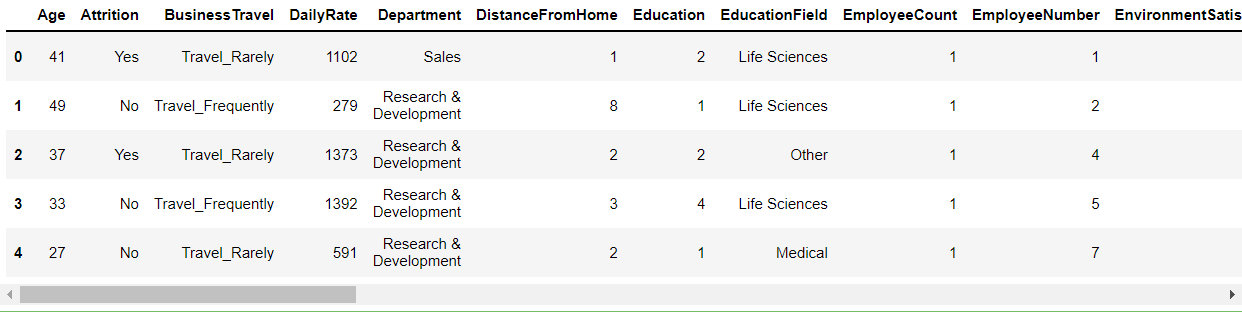
print(f'Number of rows and columns in given Data Frame is {df.shape}')



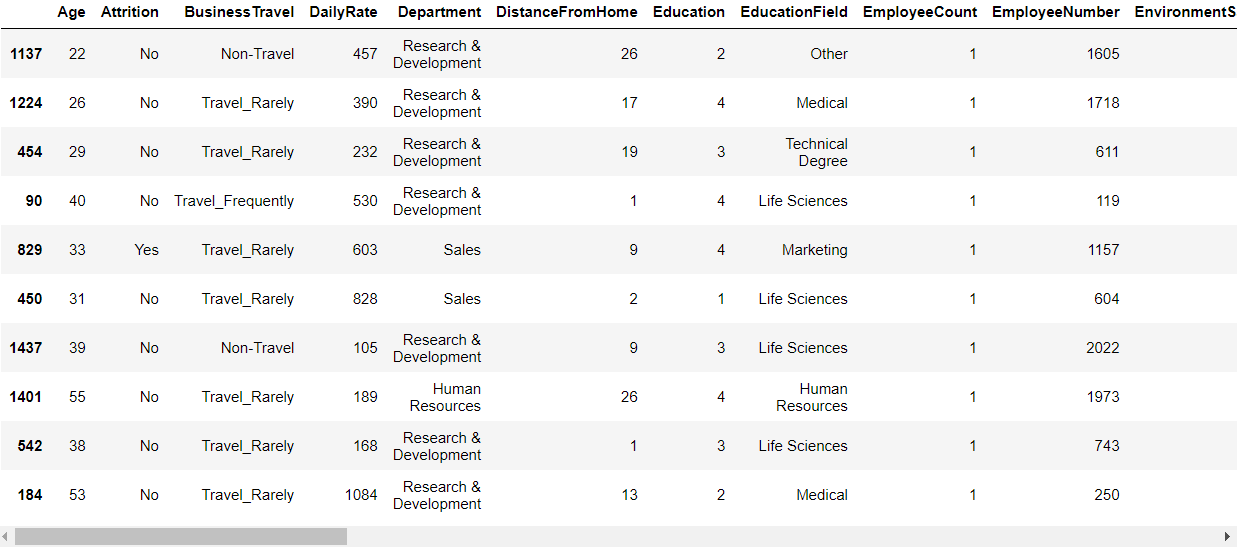
df.shape



df.head()



df.sample(10)



**Observation -**

1. There are total 1470 records with 35 columns in each entry.

### Checking Duplicate Values

# Removing duplicate values

df.duplicated().sum()

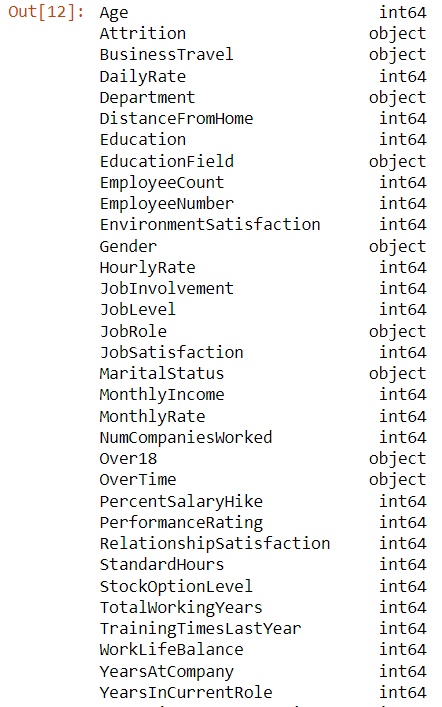


**Observation -**

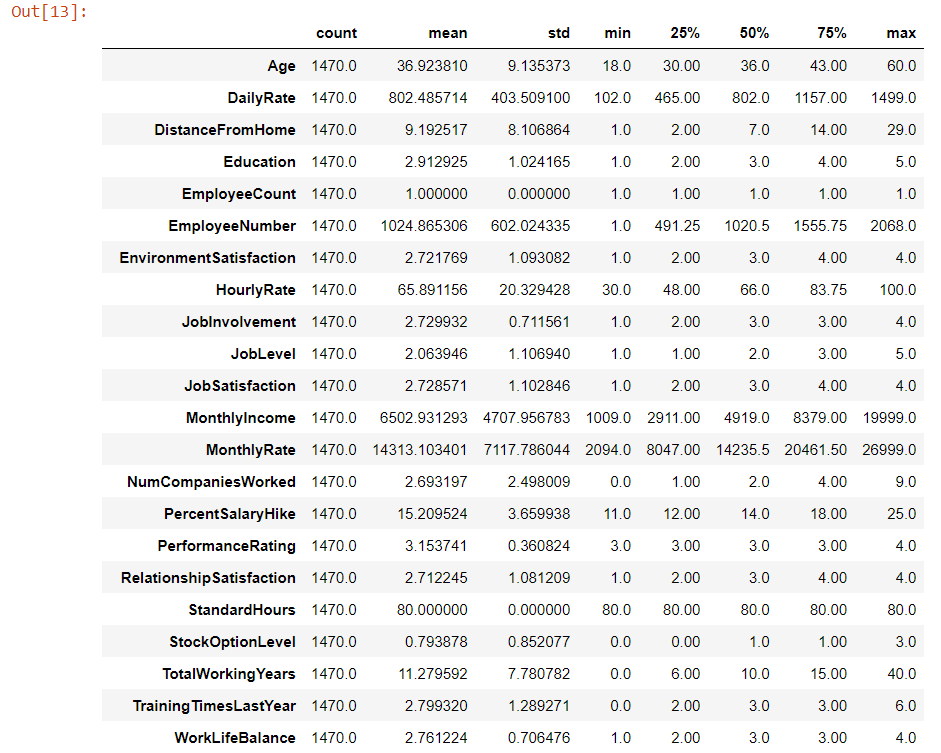
1. There are no duplicate value

### Five Number Summery for Numerical Data

df.dtypes



df.describe().T



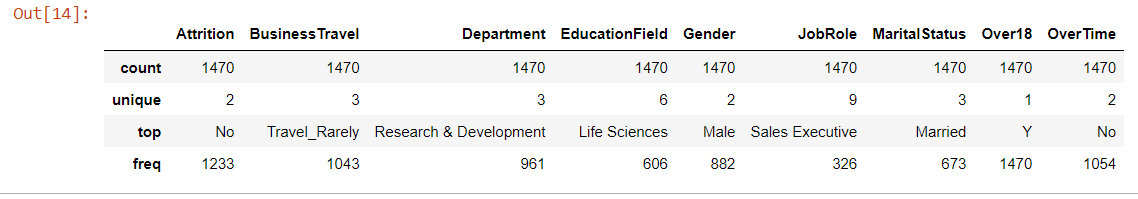
**Observations -**

Looking at count column, there are no missing values. There may be outliers, but no missing values in continuous data

1. Age: Average age of employees is 36
2. DailyRate: Average daily rate is 802.49 with min. rate 102 and max. rate is 1499
3. DistanceFromHome:
   * For 50% of employees have office within 7 km.
   * Average distance employee needs to travel to reach company is 9.19
4. Education: Education of employees
5. EmployeeCount: Employee count as 1
6. EmployeeNumber: Employee Number
7. EnvironmentSatisfaction:
   * Average employee satisfaction is 2.72
   * 50% of employee has satisfaction level less than 3
   * 25% of employee has satisfaction level less than 2
8. HourlyRate: Average hourly rate of employee is 65.89
9. JobInvolvement: Average 2.729 with min 1 and max 4
10. JobLevel: 50% of employees has job level 1 and 2
11. JobSatisfaction: Average job satisfaction is 2.72
12. MonthlyIncome:
    * Average monthly income is 6502.93
    * 50 % of employees has income less than 4919.0
13. MonthlyRate: Average monthly rate of employee is 14313.103401
14. NumCompaniesWorked: Record having employees with 0 to 9 companies
15. PercentSalaryHike:
    * Average salary hike is 15.20%
    * 50% of employees received, 14 % Salary hike
    * Minimum salary hike is 11% and maximum 25%
16. PerformanceRating:
    * All have performance rating of 3 and 4
17. RelationshipSatisfaction: Average relation satisfaction is 2.71
18. StandardHours: 80 Hours, Constant for all employees
19. StockOptionLevel: Average = 0.79, min = 1 and max = 3
20. TotalWorkingYears:
    * Average working years are 11.27
    * min is 0 and max is 40 years
    * 25% employees have experience less than 6
    * 25% employees have experience between 6 and 10
21. TrainingTimesLastYear: Total training time last year
22. WorkLifeBalance: Graded between 1 to 4 with average of 2.76
23. YearsAtCompany:
    * 25% of employee has 9 to 40 years of experience in this company
24. YearsInCurrentRole: with average of 4.229
25. YearsSinceLastPromotion: Employee get promotion in every 2 years (Average)
26. YearsWithCurrManager: 25%-25% employees have spent 0-2 and 7-17 years with current manager.

### Five Number Summery for Categorical Data

df.describe(include="O")



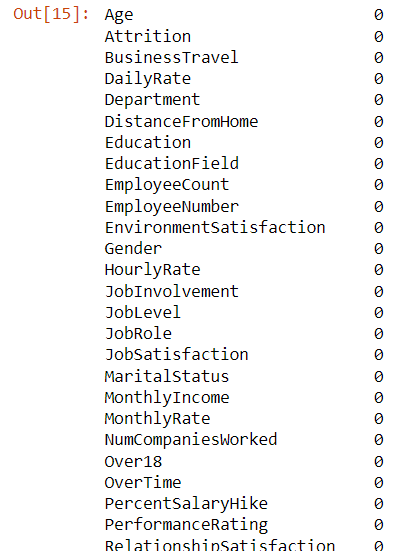
**Observations -**

1. There are less than 10 categories in each of qualitative data. This information will help us to plot various plots to understand data.
2. Attrition: There are only two categories (Yes/No): With mode of "No" having frequency of 1233, for Yes it counts as 337
3. BusinessTravel: Three category, with mode as "Travel\_Rarely" with freq. as 1043
4. Department: Three departments, with mode of "Research & Development" with frequency of 961
5. EducationField: 6 Category, with max "Life Sciences" counted 606
6. Gender: 2 categories, with mode of male with 882 counts.
7. JobRole: 9 job roles, with more sale executive (326)
8. MaritalStatus: Most of the men working are Married (673)
9. Over18: As per rule/qualification required, all employees must be above 18 years.
10. OverTime: Most of employees are not doing overtime. Only 416

### Missing Values

# Feature Wise missing Values

df.isnull().sum()



# All missing Values

print("There are total", df.isnull().sum().sum(), "missing values in dataset")



## EDA Concluding Remarks

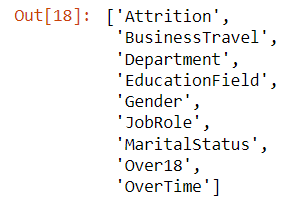
### Separating Data as Categorical and Numerical

### Separating Data as categorical and Numerical Data

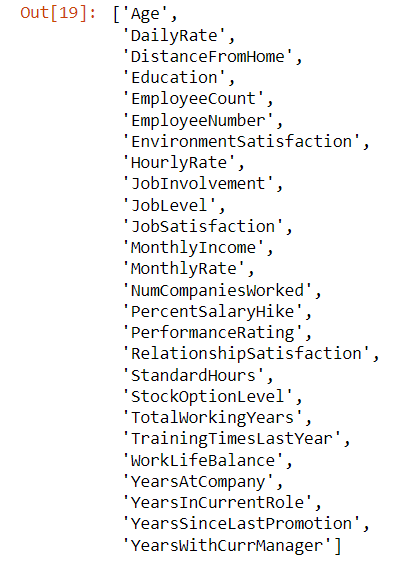
categorical\_features = [feature for feature in df.columns if df[feature].dtype == "O"]

numerical\_features = [feature for feature in df.columns if df[feature].dtype != "O"]

print(categorical\_features)



print(numerical\_features)



print(f'Total Categorical Features are {len(categorical\_features)}')

print(f'Total Numerical Features are {len(numerical\_features)}')

****

**Observation -**

1. Total Categorical features are 9.
2. Total Numerical Features are 26.

### Count plots with Hue as Attrition

#### BusinessTravel

# Countplot for BusinessTravel

plt.figure(dpi=100)

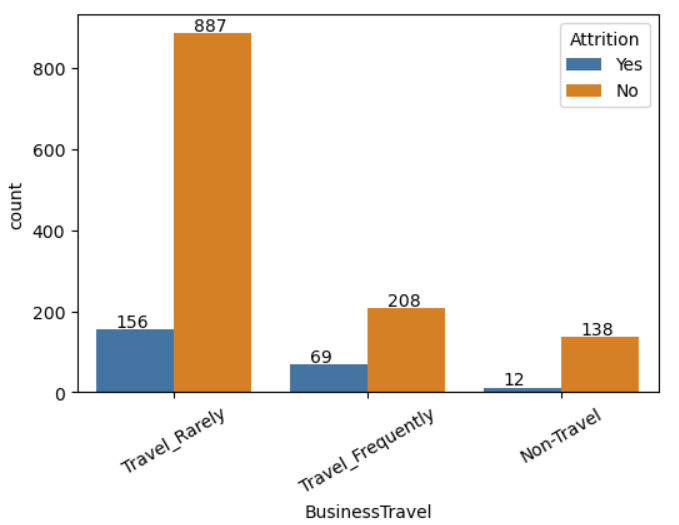
plot = sns.countplot('BusinessTravel',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.BusinessTravel.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

* Employee who travel frequently have more chances of departure (Almost 25%)
* Distribution as
  + Travel\_Rarely = 1043
  + Travel\_Frequently = 277
  + Non-Travel = 150

#### Department

# Countplot for

plt.figure(dpi=100)

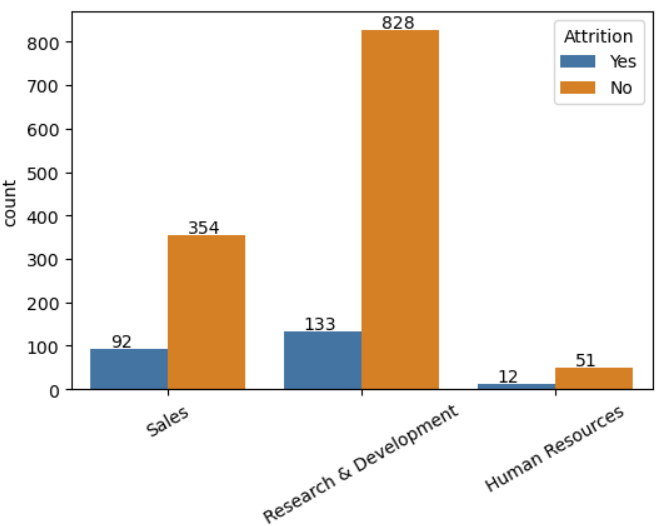
plot = sns.countplot('Department',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.Department.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

* Almost 33% of employees working in Sales department departed
* HR and R&D people are less likely to depart
* Value Count for Each Department as
  + Research & Development = 961
  + Sales = 446
  + Human Resources = 63

#### EducationField

# Countplot for Education Field

plt.figure(dpi=100)

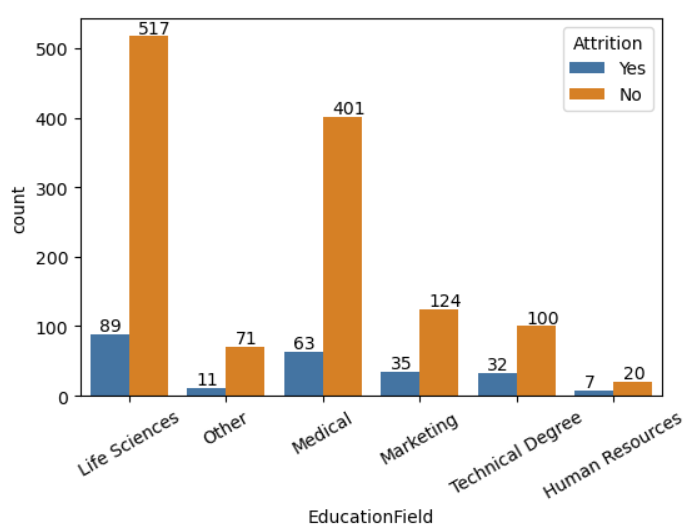
plot = sns.countplot('EducationField',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.EducationField.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

* Most of the employees are from Life Sciences and Medical Field, Almost 1070
* Attrition happens for almost every field. But for Marketing, HR and Technical Degree people probability seems higher.

#### Gender

# Countplot for Gender

plt.figure(dpi=100)

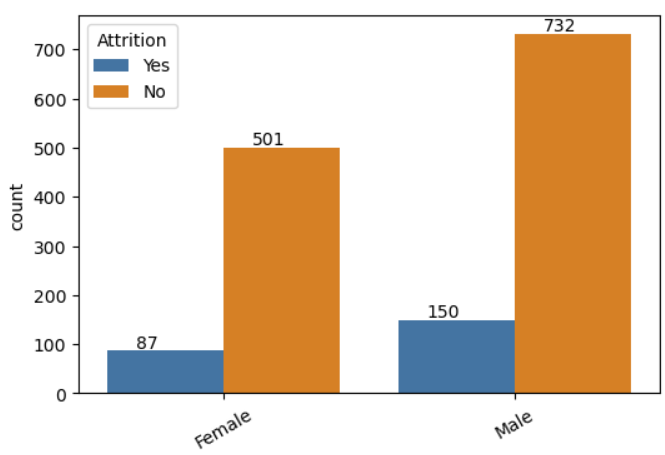
plot = sns.countplot('Gender',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.Gender.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



print(f'% of Men’s Departed = {round((150/882)\*100, 2)} %')

print(f'% of Females Departed = {round((87/588)\*100, 2)} %')

**Observation -**

* Almost attrition rate is same for both men’s and women. There is slight difference but can be treated as same (Diff 2.8)
  + % Of Men’s Departed = 17.01 %
  + % Of Females Departed = 14.8 %

#### JobRole

# Countplot for JobRole

plt.figure(dpi=100, figsize=(20, 8))

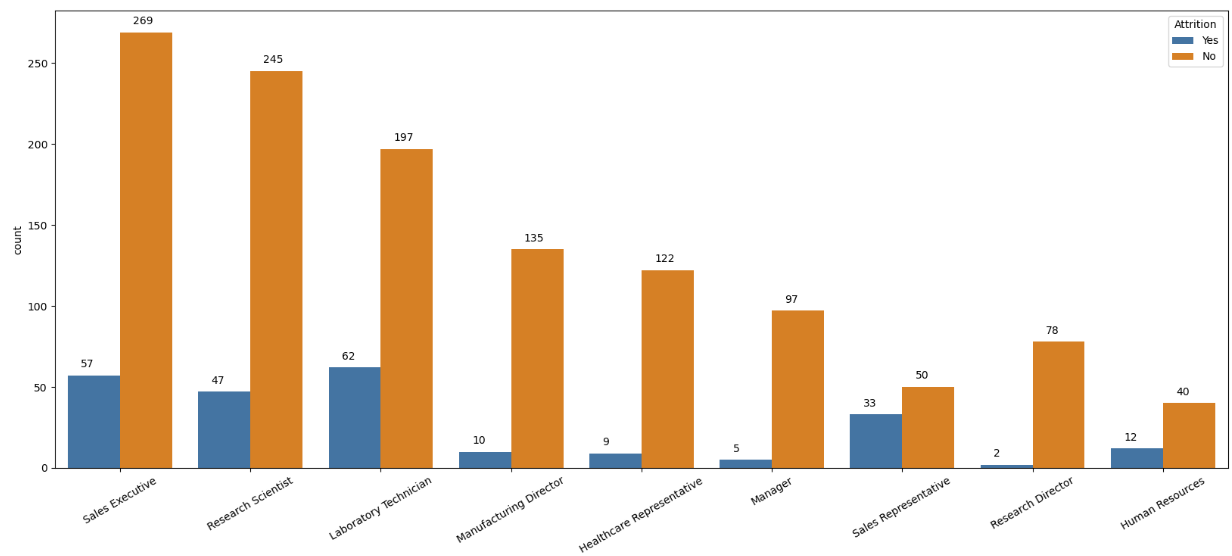
plot = sns.countplot('JobRole',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.JobRole.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

Sales Representative are more likely to depart

* Lesser chance of departure - Managers, Research Director, Healthcare Representative
* Higher chance of departure - Research Scientist, Sales Executive, HR, Laboratory Technician

#### MaritalStatus

# Countplot for Marital Status

plt.figure(dpi=100)

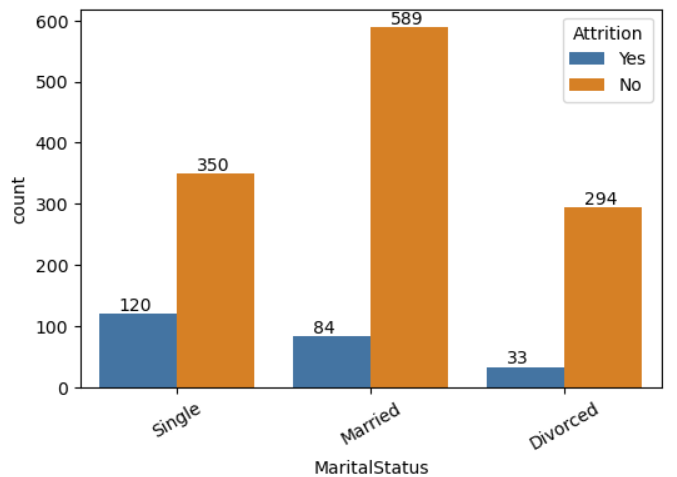
plot = sns.countplot('MaritalStatus',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.MaritalStatus.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

Chances of leaving is more for singles

* 25% of singles has departed
* Only 12% of married people opted departure from current organisation
* Only 10% of Divorced people opted departure from current organisation

#### Overtime

# Countplot for OverTime

plt.figure(dpi=100)

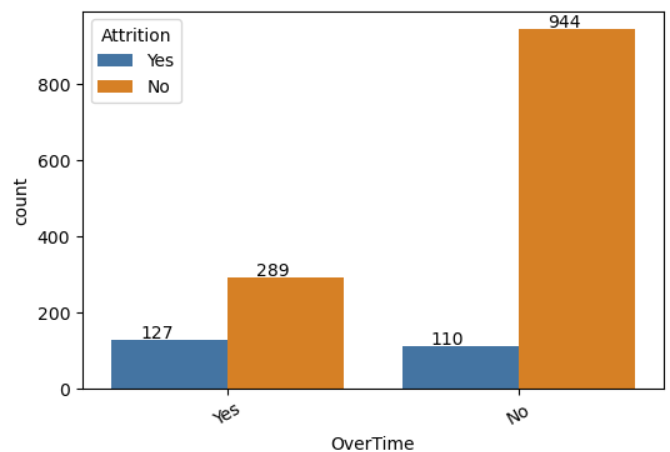
plot = sns.countplot('OverTime',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.OverTime.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

* 30% of employees doing over time has more chances of leaving company
* Out of rest of the employees only 10% of employees left company

#### Age

# Countplot for OverTime

plt.figure(dpi=100, figsize=(20,8))

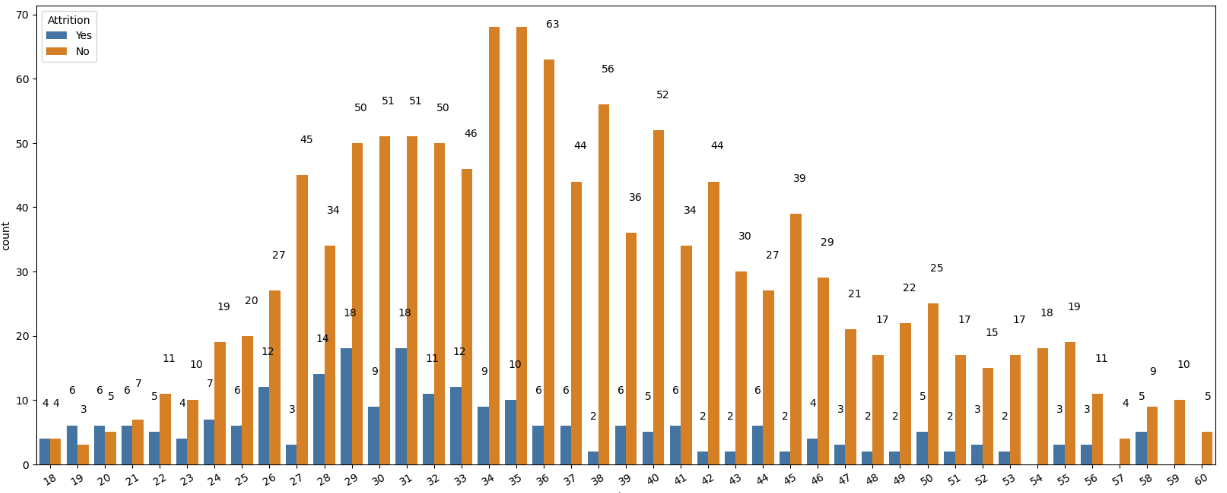
plot = sns.countplot('Age',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.Age.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

* Employees with age ranging from 18-21 has 50% of chance of departing.
* Employees with age ranging from 22-32 has 20-25% of chance of departing.
* As Age of employee increases, chances of departing decreases.

#### DistanceFromHome

# Countplot for DistanceFromHome

plt.figure(dpi=100, figsize=(20,8))

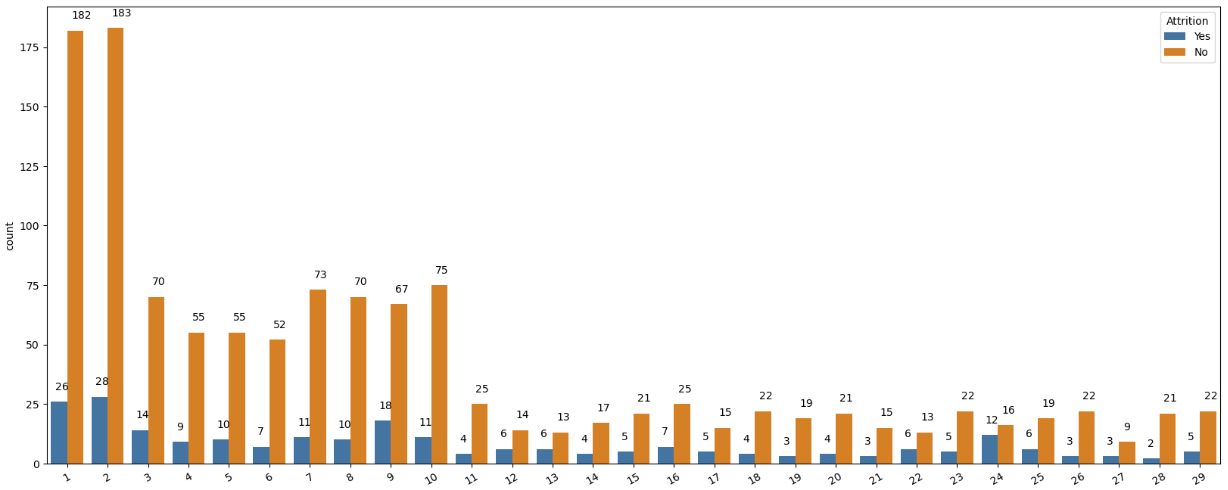
plot = sns.countplot('DistanceFromHome',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.DistanceFromHome.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

* It seems no effect of distance on attrition, as % of getting attrition is ranging between 10-20 % for all.

#### Education

# Countplot for Education

plt.figure(dpi=100, figsize=(20,8))

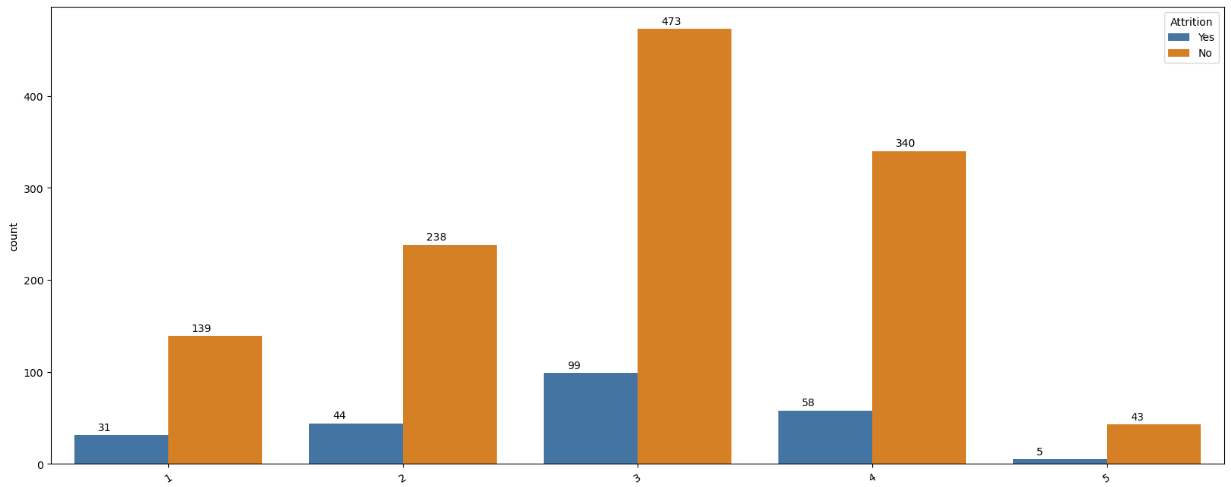
plot = sns.countplot('Education',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.Education.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

* It seems no effect of education on attrition, as % of getting attrition is ranging between 10-20 % for all.

#### Environment Satisfaction

# Countplot for EnvironmentSatisfaction

plt.figure(dpi=100, figsize=(20,8))

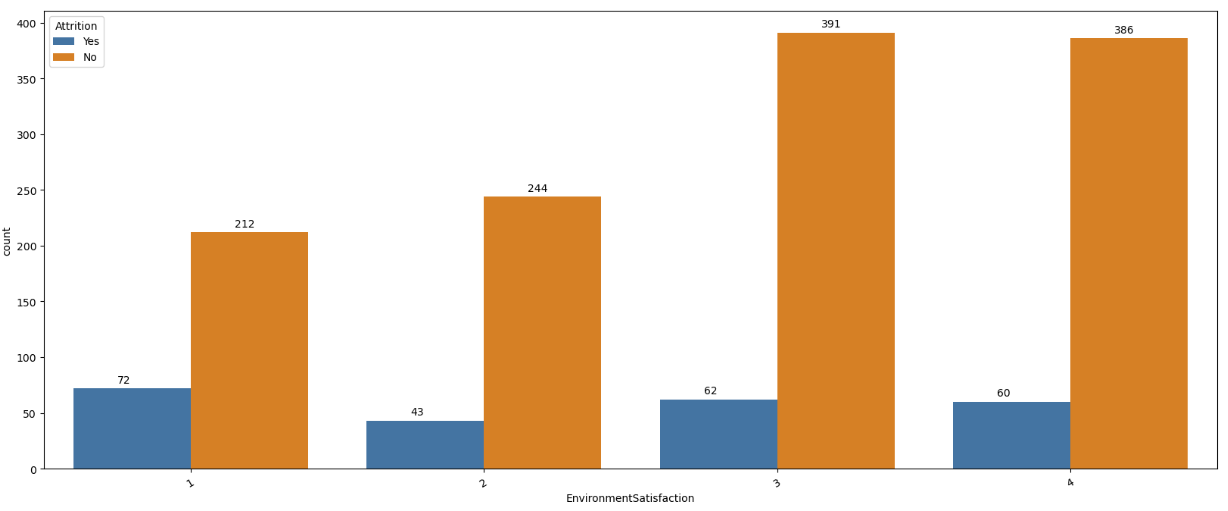
plot = sns.countplot('EnvironmentSatisfaction',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.EnvironmentSatisfaction.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.15, p.get\_height()+5))



**Observation -**

* Employees having 1 Environment Satisfaction are more likely to leave company

#### JobInvolvement

# Countplot for JobInvolvement

plt.figure(dpi=100, figsize=(20,8))

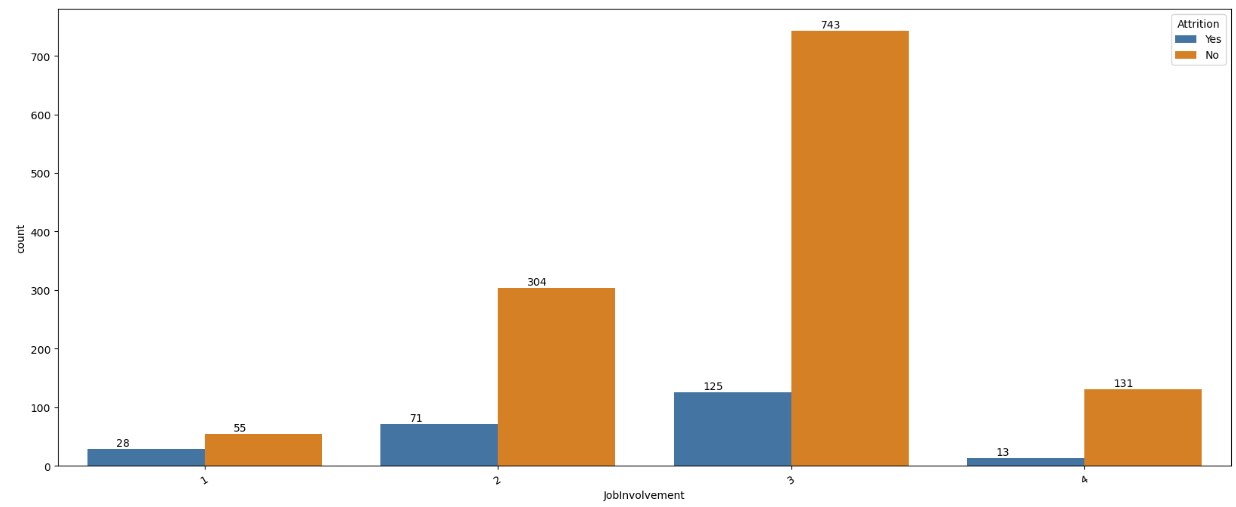
plot = sns.countplot('JobInvolvement',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.JobInvolvement.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

* With JobInvolment=1, 33% Chance of leaving company
* With JobInvolment=2, 18% Chance of leaving company
* With JobInvolment=3, 15% Chance of leaving company
* With JobInvolment=4, 8% Chance of leaving company

#### JobLevel

# Countplot for JobLevel

plt.figure(dpi=100, figsize=(20,8))

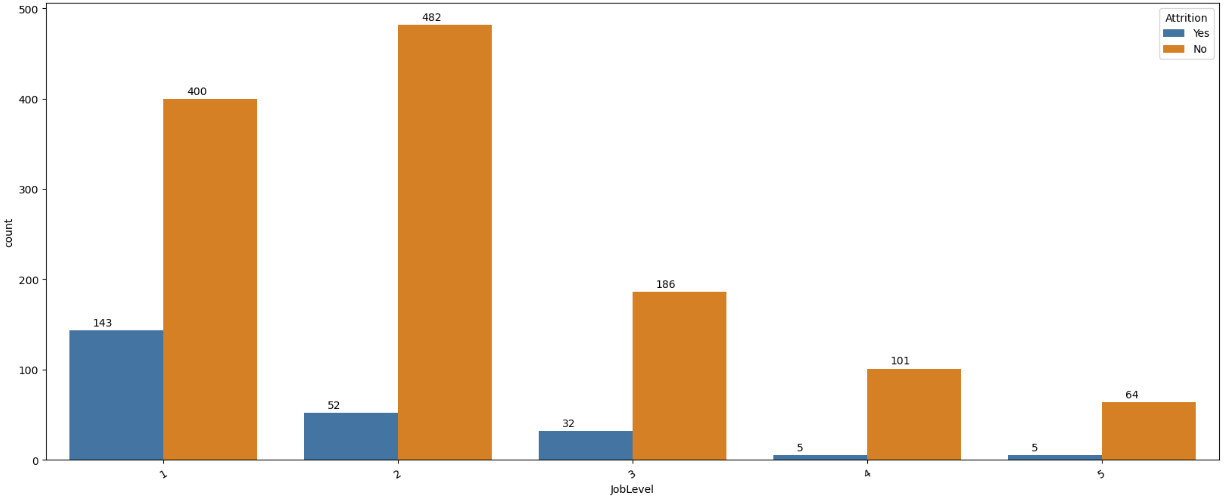
plot = sns.countplot('JobLevel',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.JobLevel.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

* For jobLevel = 1, chances employee leaving company are more around 25 %

#### JobSatisfaction

# Countplot for JobSatisfaction

plt.figure(dpi=100, figsize=(20,8))

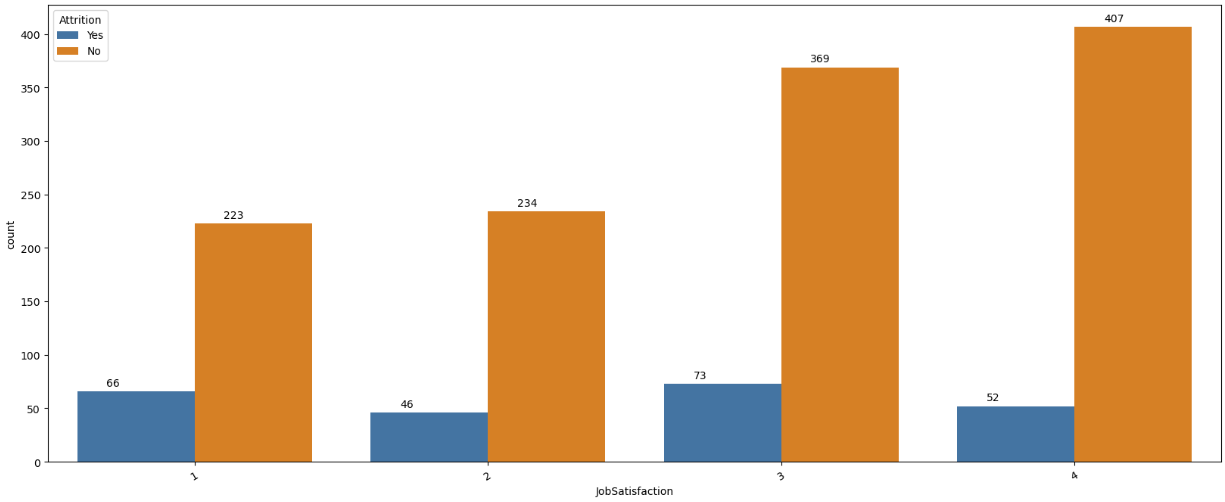
plot = sns.countplot('JobSatisfaction',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.JobSatisfaction.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

* Employees with less satisfaction level are more likely to depart in another organisation

#### NumCompaniesWorked

# Countplot for NumCompaniesWorked

plt.figure(dpi=100, figsize=(20,8))

plot = sns.countplot('NumCompaniesWorked',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.NumCompaniesWorked.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))

**Observation -**

* Employees who worked in more companies are more likely to depart
* Employees with less worked companies has low chance of leaving company

#### PercentSalaryHike

# Countplot for PercentSalaryHike

plt.figure(dpi=100, figsize=(20,8))

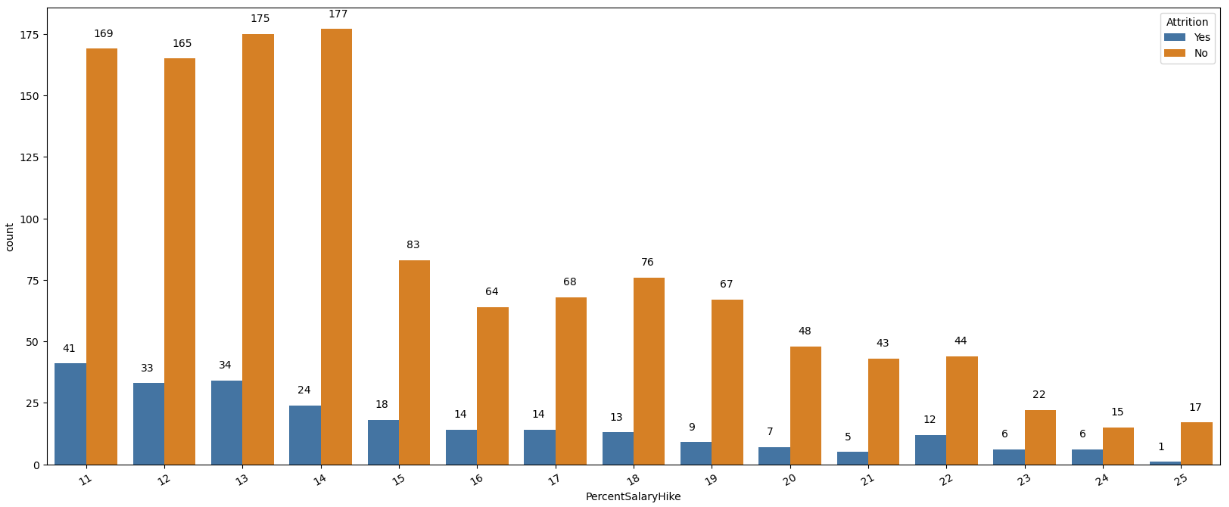
plot = sns.countplot('PercentSalaryHike',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.PercentSalaryHike.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observations -**

* It is difficult to establish relationship based only on % hike, because employees with high salary hike also more tends to leave company (Around 25% chance).
* Also, employees with % hike 11-15 are more likely to leave company (15% Chance)

#### PerformanceRating

# Countplot for PerformanceRating

plt.figure(dpi=100)

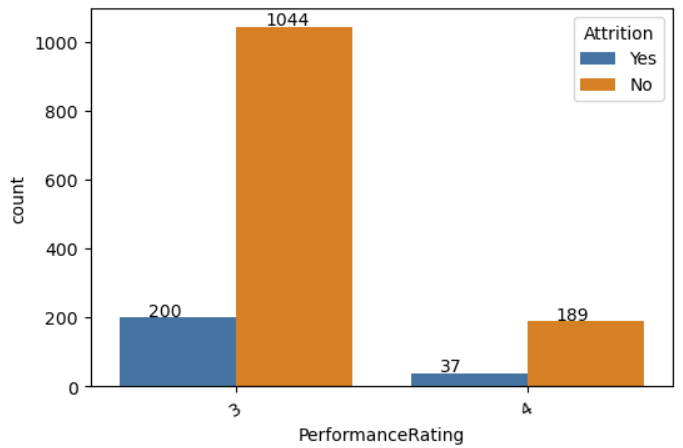
plot = sns.countplot('PerformanceRating',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.PerformanceRating.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

* Both ratings have almost same departing %

#### RelationshipSatisfaction

# Countplot for RelationshipSatisfaction

plt.figure(dpi=100)

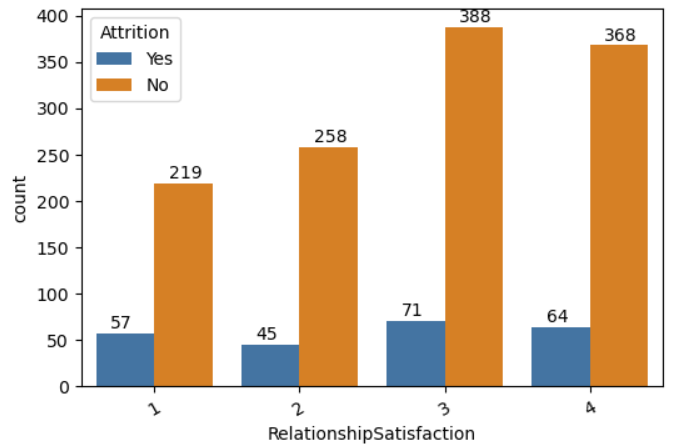
plot = sns.countplot('RelationshipSatisfaction',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.RelationshipSatisfaction.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

* Employees with more Relationship Satisfaction are more likely to stay.

#### StockOptionLevel

# Countplot for StockOptionLevel

plt.figure(dpi=100)

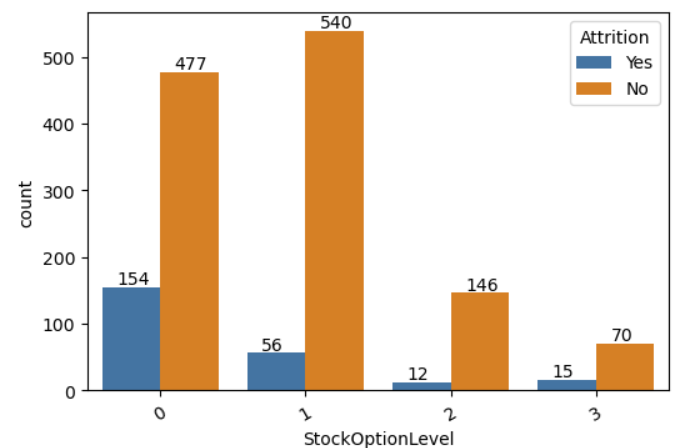
plot = sns.countplot('StockOptionLevel',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.StockOptionLevel.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation** -

* Employee with more StockOptionLevel are more likely to stay in organisation

#### TotalWorkingYears

# Countplot for TotalWorkingYears

plt.figure(dpi=100, figsize=(20,8))

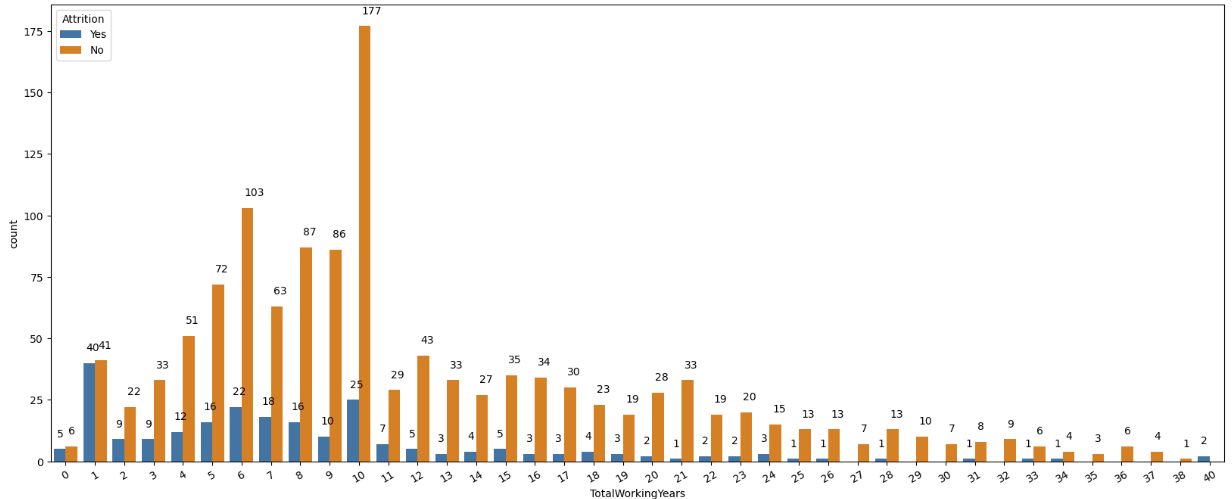
plot = sns.countplot('TotalWorkingYears',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.TotalWorkingYears.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



**Observation -**

* Employees with total working years ranging between 0-10 are more likely to leave company.

#### TrainingTimesLastYear

# Countplot for TrainingTimesLastYear

plt.figure(dpi=100, figsize=(20,8))

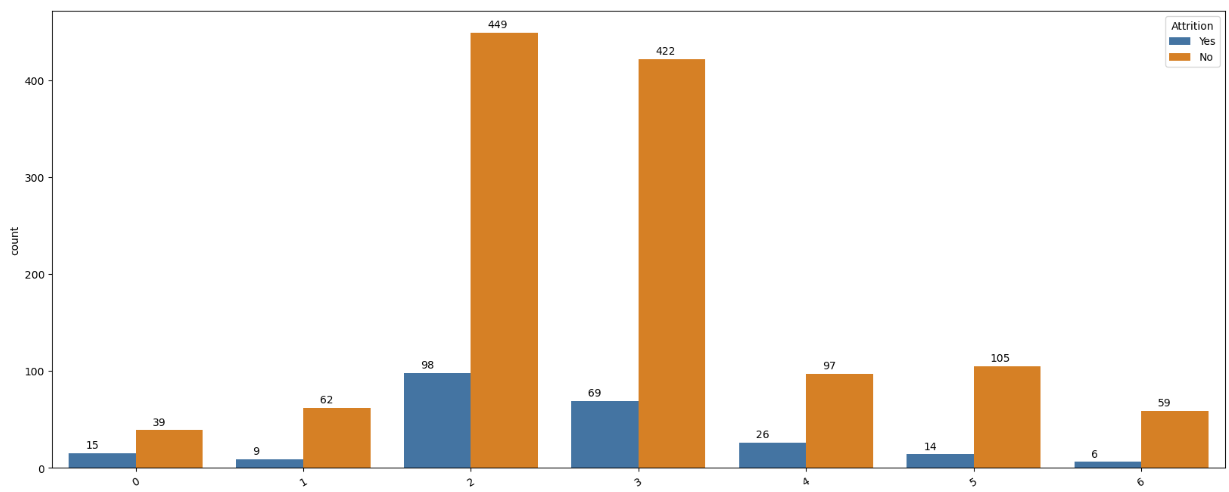
plot = sns.countplot('TrainingTimesLastYear',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.TrainingTimesLastYear.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



# Countplot for WorkLifeBalance

plt.figure(dpi=100)

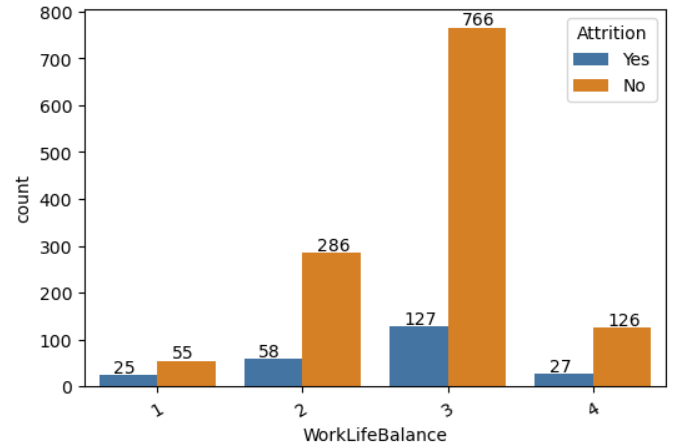
plot = sns.countplot('WorkLifeBalance',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.WorkLifeBalance.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



# Countplot for YearsAtCompany

plt.figure(dpi=100, figsize=(20,8))

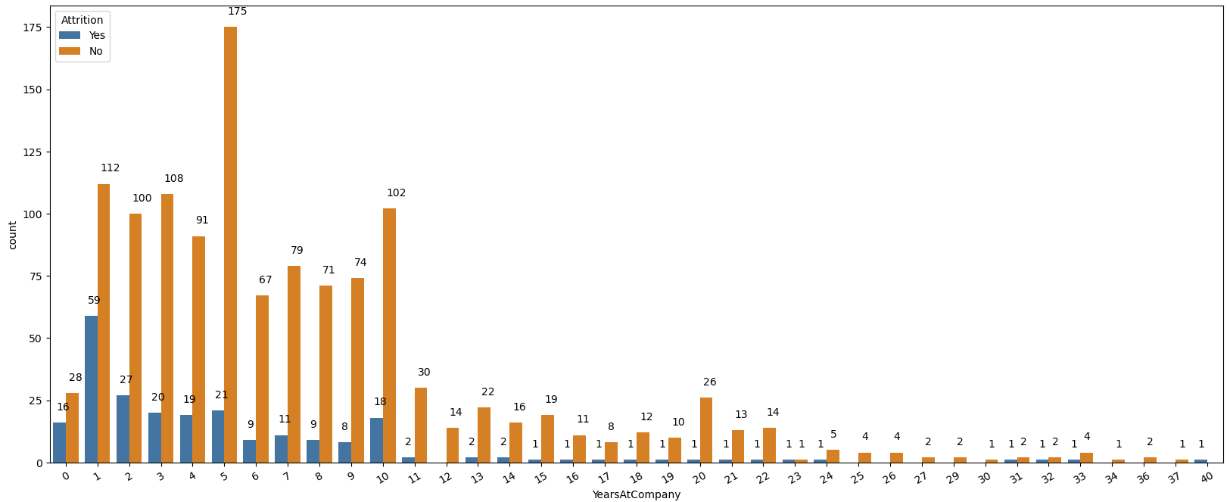
plot = sns.countplot('YearsAtCompany',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.YearsAtCompany.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



# Countplot for YearsInCurrentRole

plt.figure(dpi=100, figsize=(20,8))

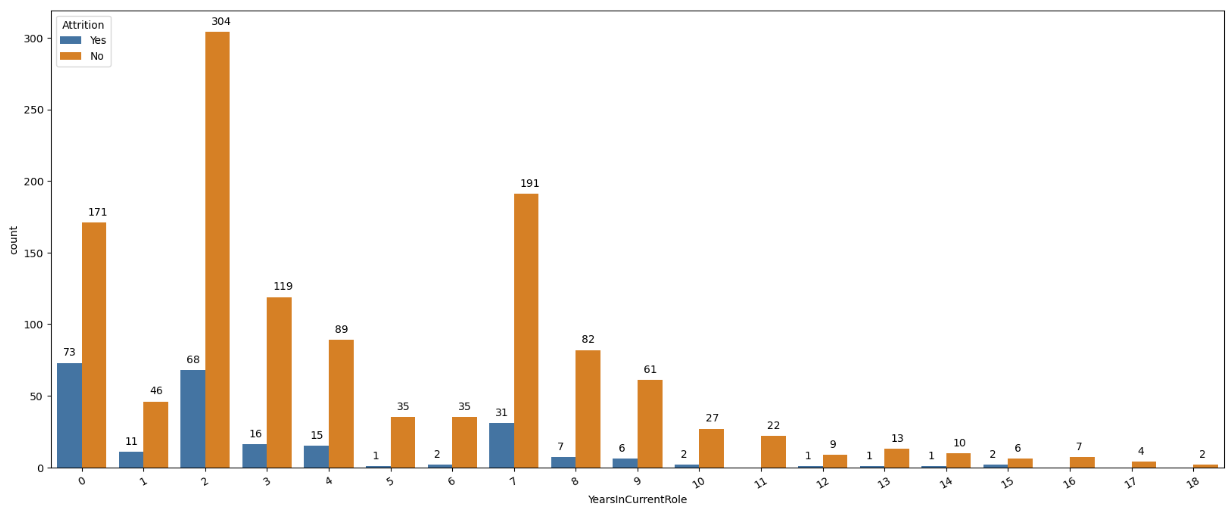
plot = sns.countplot('YearsInCurrentRole',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.YearsInCurrentRole.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



# Countplot for YearsSinceLastPromotion

plt.figure(dpi=100, figsize=(20,8))

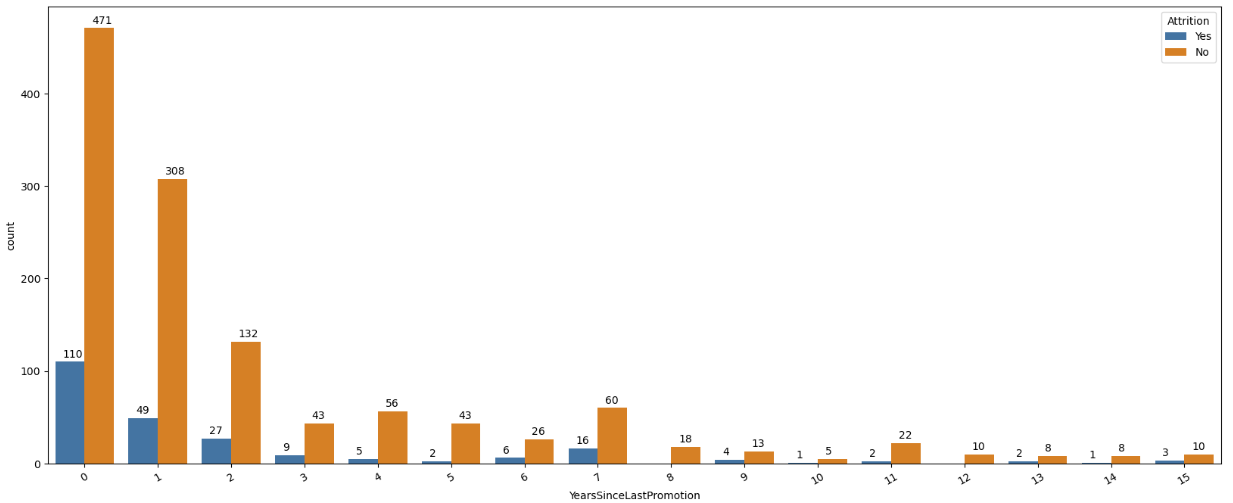
plot = sns.countplot('YearsSinceLastPromotion',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.YearsSinceLastPromotion.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



# Countplot for YearsWithCurrManager

plt.figure(dpi=100, figsize=(20,8))

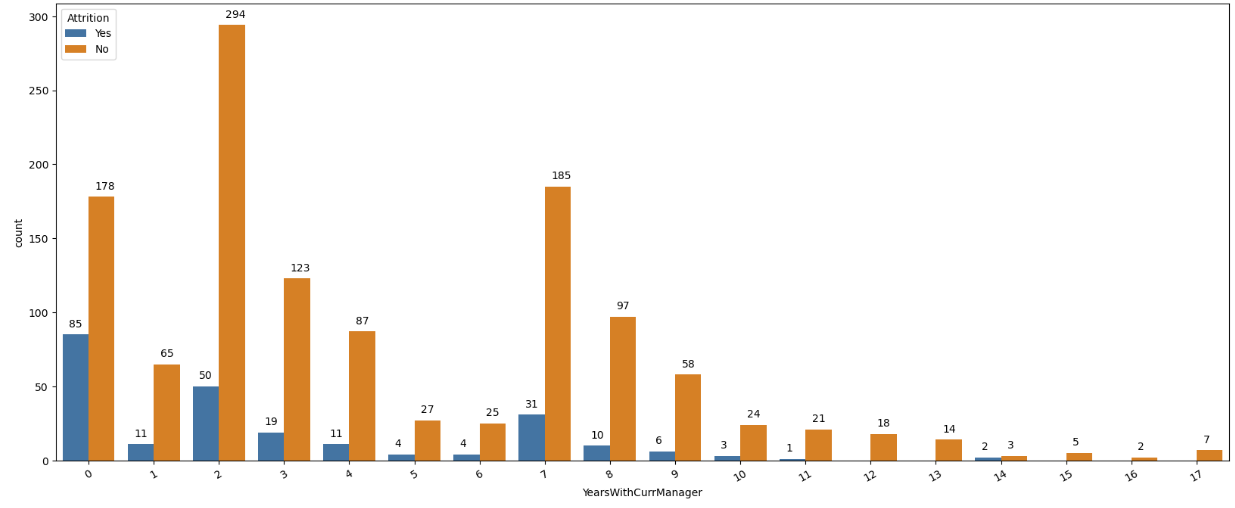
plot = sns.countplot('YearsWithCurrManager',data=df, hue='Attrition')

plot.set\_xticklabels(plot.get\_xticklabels(),rotation = 30)

print(df.YearsWithCurrManager.value\_counts())

for p in plot.patches:

plot.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.1, p.get\_height()+5))



#### Observation -

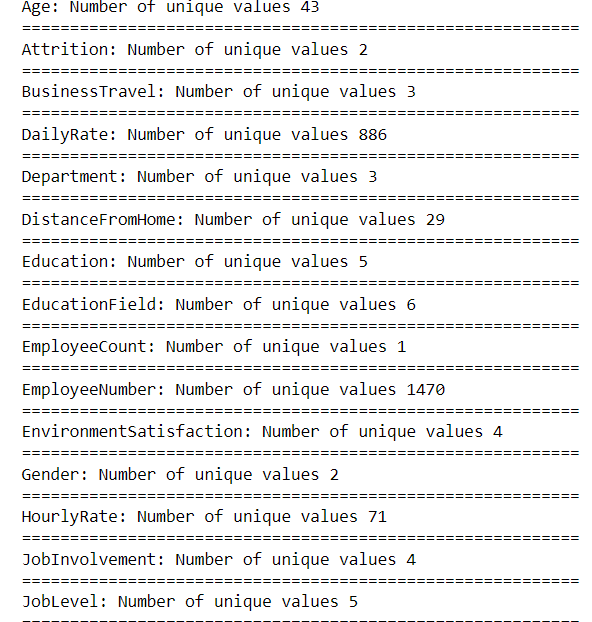
1. TrainingTimesLastYear
   * Employees getting training 2-3 times in year
   * TrainingTimes doesn’t affect departing status, as it is same for all categories
2. WorkLifeBalance
   * Employees with 1 work life balance are more likely to leave company
3. YearsAtCompany
   * Employees with 0-10 years at company are more likely to leave company.
4. YearsInCurrentRole
   * Employees with more years in current role are less likely to leave company
5. YearsSinceLastPromotion
   * Employees leaving early after 1,2,3 promotion
6. YearsWithCurrManager
   * More employees tend to leaver company after working for 1-2 years.

### Checking Uniqueness of Data

for column in df.columns:

print(f"{column}: Number of unique values {df[column].nunique()}")

print("==========================================================")



**Observation**

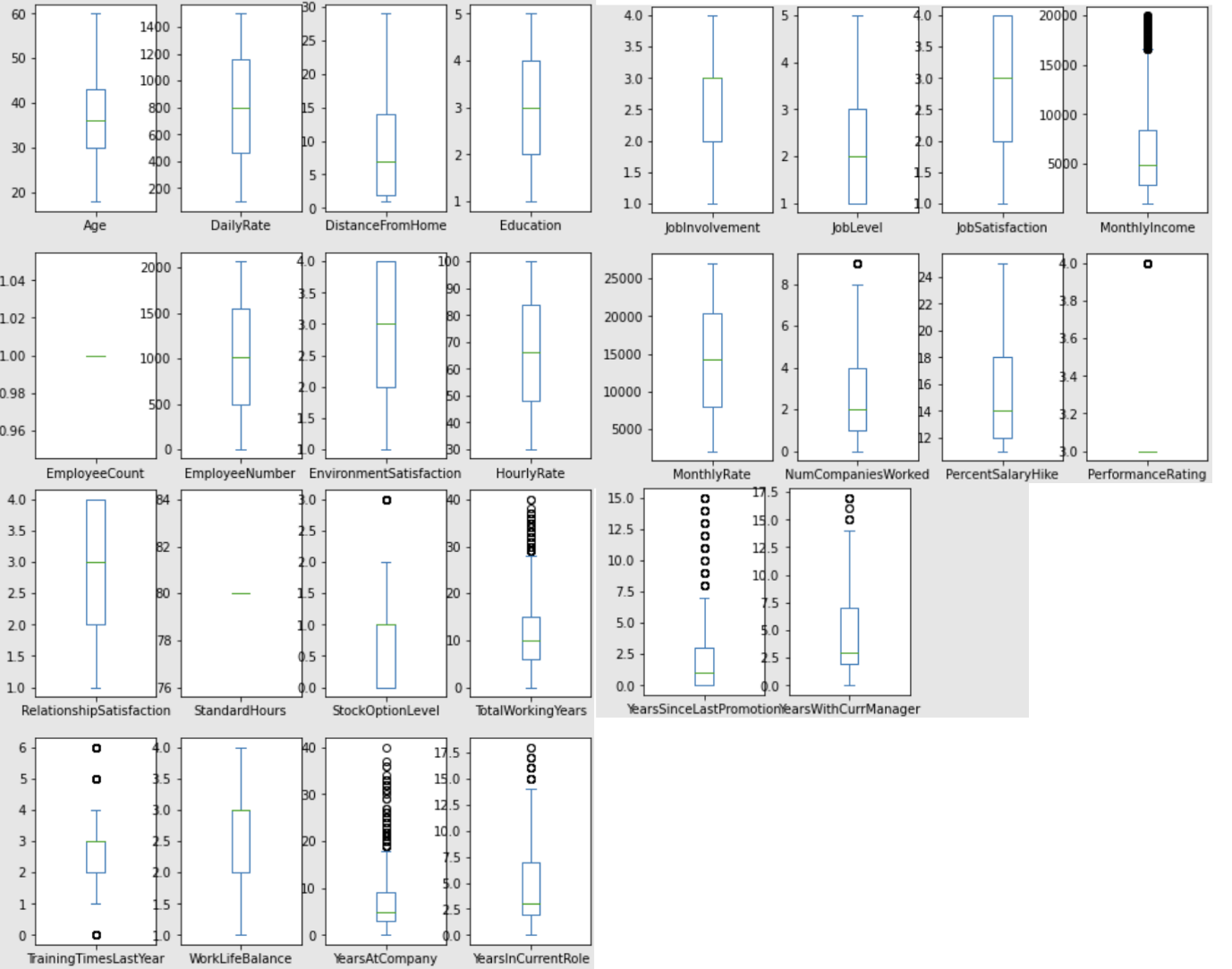
* 'EmployeeCount', 'Over18' and 'StandardHours' have only one unique values
* 'EmployeeNumber' has 1470 unique values

Above feature aren't playing important role, so considering dropping them.

### Checking and Handling Outliers

df.plot(kind='box',subplots=True,figsize=(8,25),layout=(7,4))

# Plotting individual graph for better understanding of graph



**Observation -**

Outliers presents in

* MonthlyIncome
* NumCompaniesWorked
* PerformanceRating
* TotalWorkingYears
* TrainingTimesLastYear
* YearsAtCompany
* YearsInCurrentRole
* YearsSinceLastPromotion
* YearsWithCurrManager

cols\_with\_outliers = ['MonthlyIncome','NumCompaniesWorked', 'PerformanceRating', 'TotalWorkingYears', 'TrainingTimesLastYear', 'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion', 'YearsWithCurrManager']

plt.figure(figsize = (10, 10), dpi=250)

for i in range(0, len(cols\_with\_outliers)):

plt.subplot(5, 4, i+1)

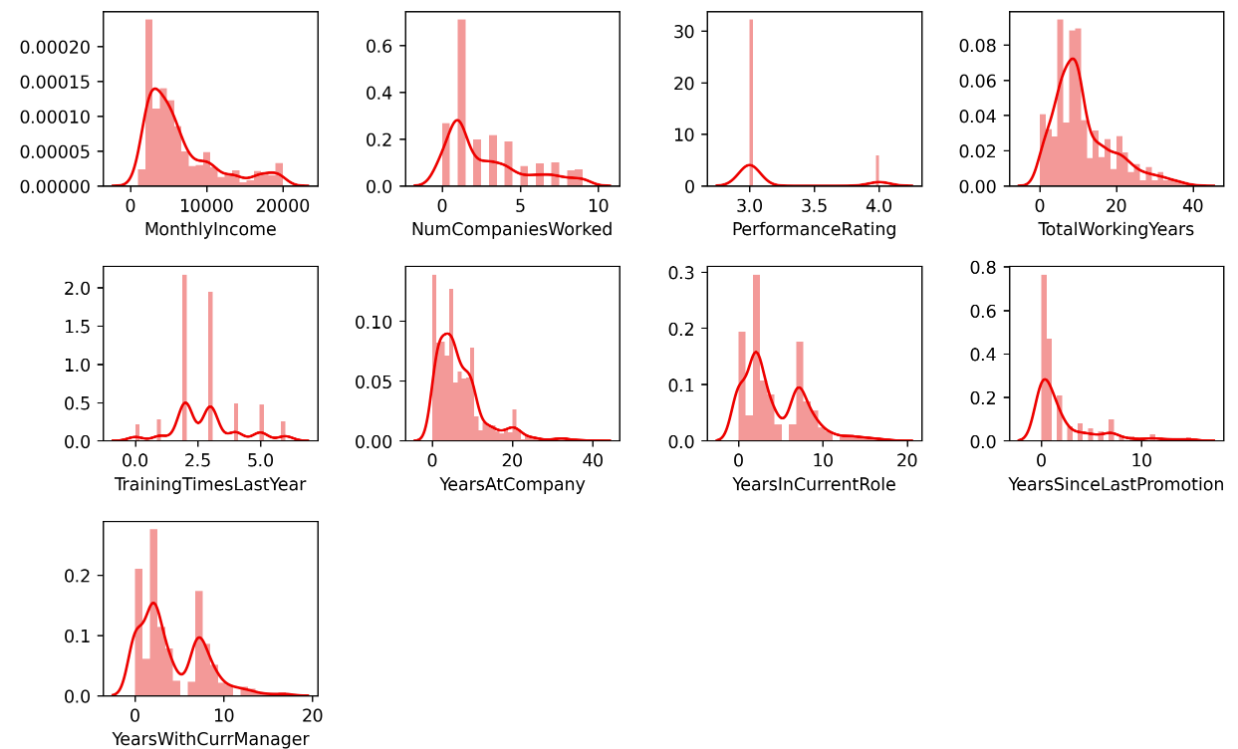
sns.distplot(df[cols\_with\_outliers[i]], color = 'red')

plt.ylabel('')

# plt.suptitle('Outliers Before Cleansing', y = 0.1)

plt.tight\_layout(pad = 1.5)

plt.show()



df2 = df.copy()

for feature in cols\_with\_outliers:

IQR = df2[feature].quantile(0.75) - df2[feature].quantile(0.25)

upper\_bond = df2[feature].quantile(0.75) + (IQR \* 1.5)

lower\_bond = df2[feature].quantile(0.25) - (IQR \* 1.5)

df2[feature] = np.where(df2[feature]>upper\_bond,upper\_bond,df2[feature])

df2[feature] = np.where(df2[feature]<lower\_bond,lower\_bond,df2[feature])

plt.figure(figsize = (12, 12), dpi=250)

for i in range(0, len(cols\_with\_outliers)):

plt.subplot(8, 4, i+1)

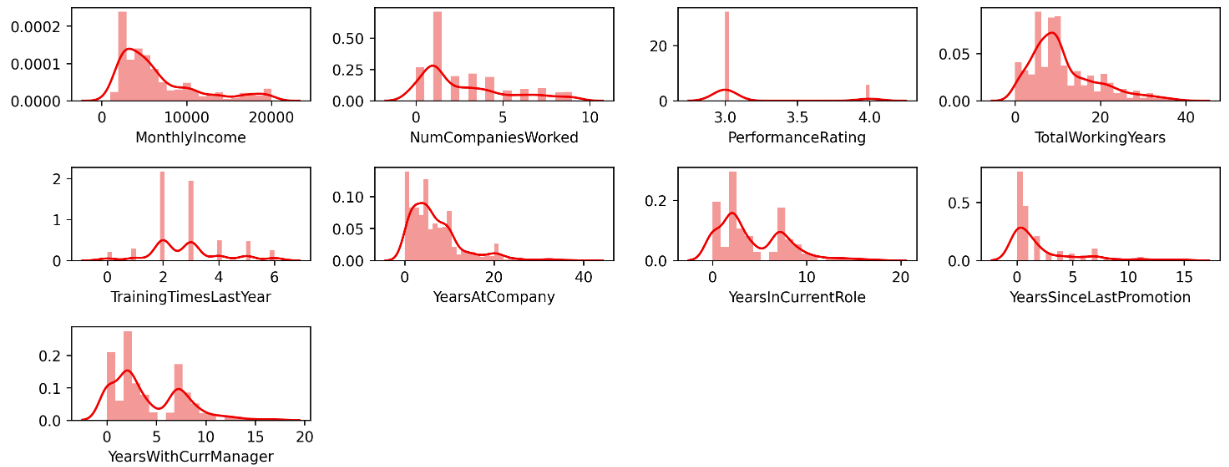
sns.distplot(df[cols\_with\_outliers[i]], color = 'red')

plt.ylabel('')

# plt.suptitle('Outliers Before Cleansing', y = 0.05)

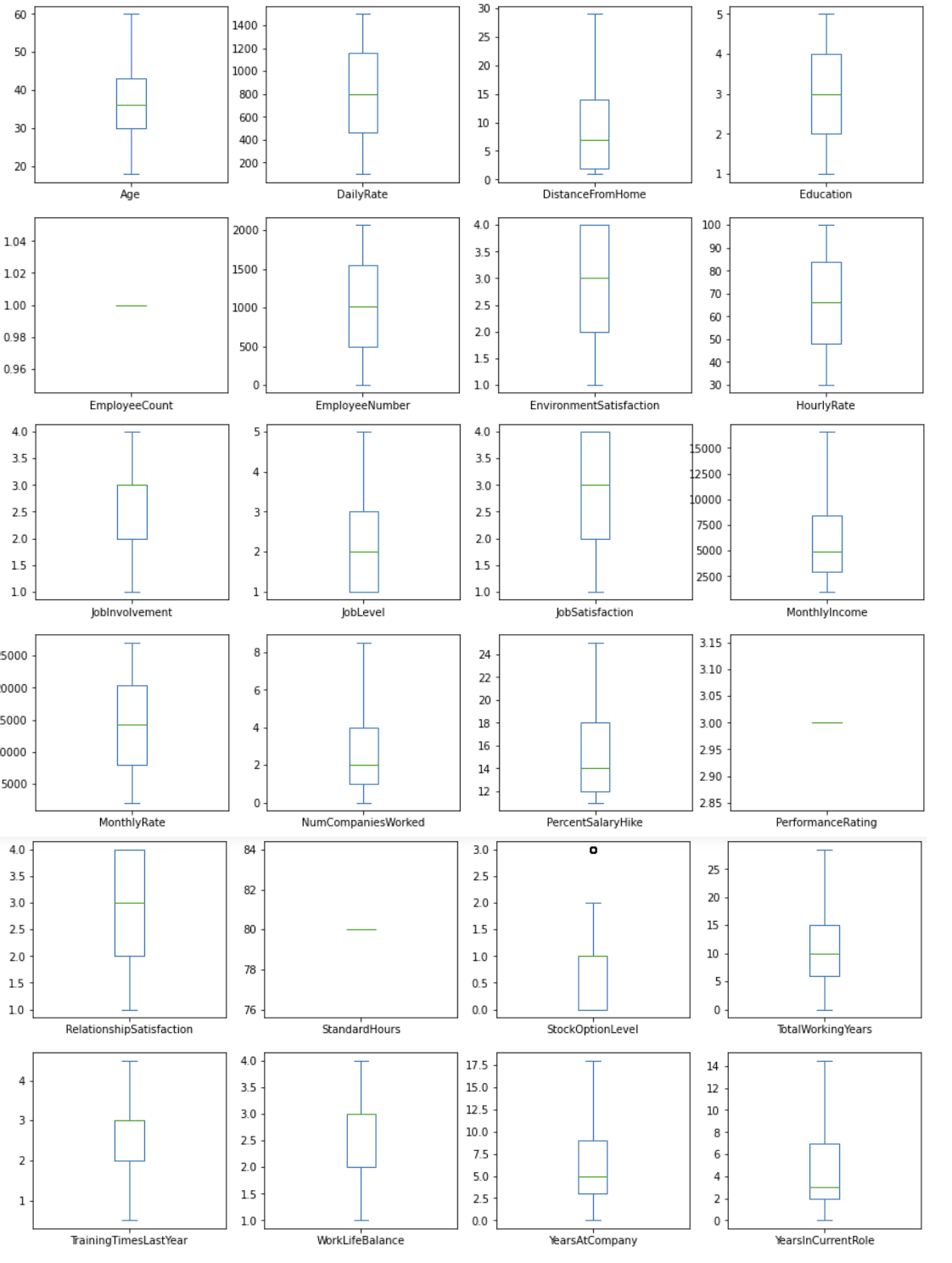
plt.tight\_layout(pad = 1.5)

plt.show()



df2.plot(kind='box',subplots=True,figsize=(15,25),layout=(7,4))

# plotting individual graph for better understanding of graph



df2.shape

**Observation -**

* Based on Previous Observation, we can drop EmployeeCount, StandardHours, EmployeeNumber

df2.drop('EmployeeCount',axis=1,inplace=True)

df2.drop('StandardHours',axis=1,inplace=True)

df2.drop('EmployeeNumber',axis=1,inplace=True)

df2.drop('Over18',axis=1,inplace=True)

df2.shape

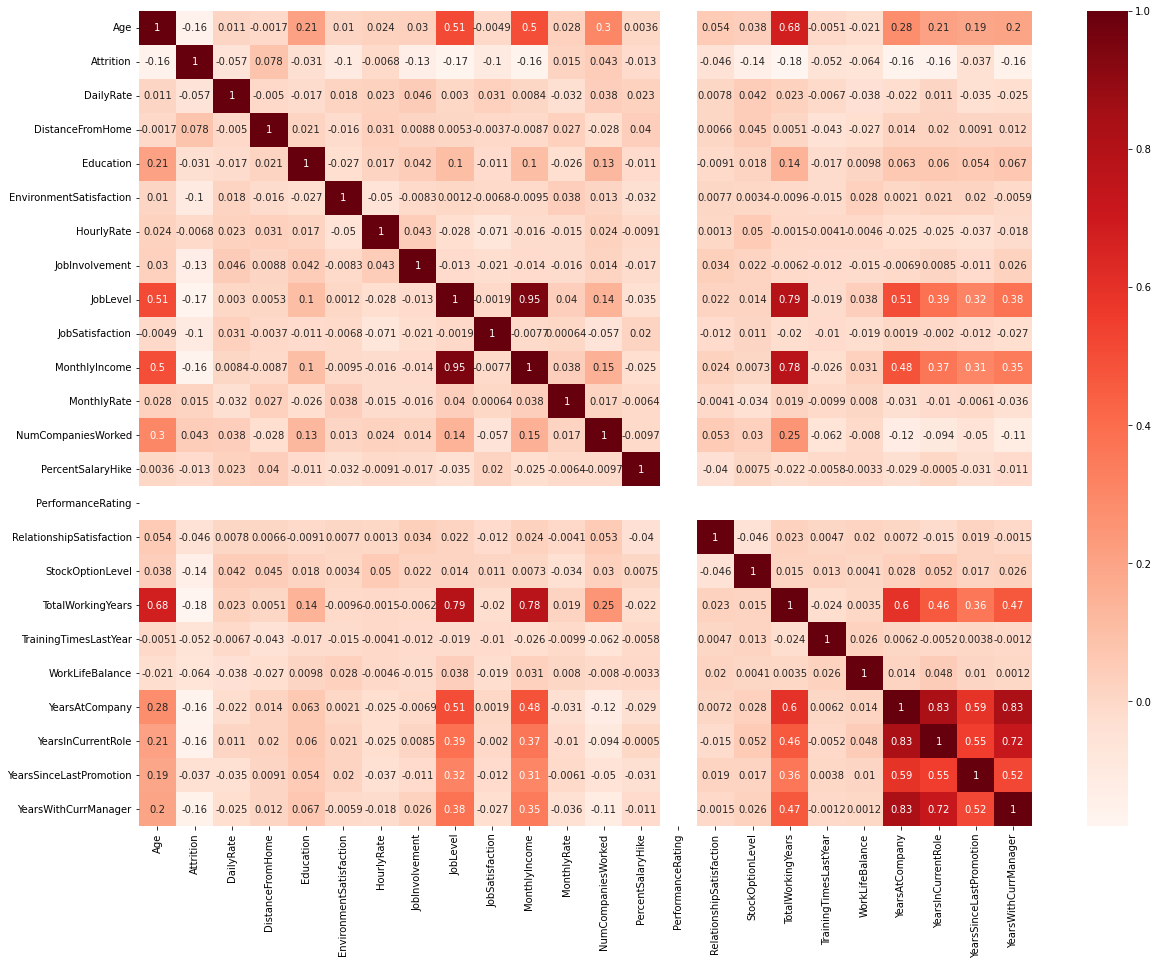
### Checking Correlation Matrix

# Changing binary representation of our target variable to 1/0

df2.Attrition = df2.Attrition.map({'Yes':1, 'No':0})

plt.figure(figsize=(20,15))

sns.heatmap(df2.corr(),cmap='Reds',annot=True)



**Observation -**

* PercentSalaryHike and PerformanceRating have a fairly strong positive relationship
* TotalWorkingYears has a fairly strong positive relationship with Age, MonthlyIncome, and JobLevel
* YearsAtCompany has a fairly strong positive relationship with YearsInCurrentRole and YearsWithCurrManager

### Removing Skewness from Data

clean\_df = df2.copy()

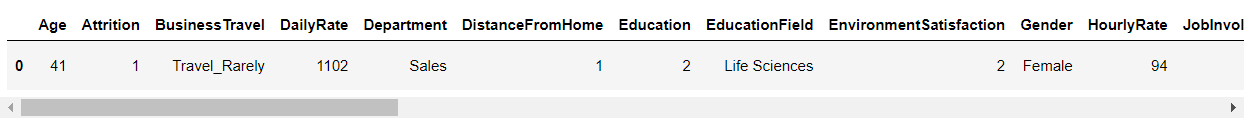
#removal of skew-ness using log function

for col in x.columns:

if x.skew().loc[col]>0.3:

x[col]=np.log1p(x[col])

clean\_df.head(1)



label\_=LabelEncoder()

#Bad becomes 0,and good becomes 1

clean\_df['BusinessTravel']=label\_.fit\_transform(clean\_df['BusinessTravel'])

clean\_df['Department']=label\_.fit\_transform(clean\_df['Department'])

clean\_df['EducationField']=label\_.fit\_transform(clean\_df['EducationField'])

clean\_df['Gender']=label\_.fit\_transform(clean\_df['Gender'])

clean\_df['JobRole']=label\_.fit\_transform(clean\_df['JobRole'])

clean\_df['MaritalStatus']=label\_.fit\_transform(clean\_df['MaritalStatus'])

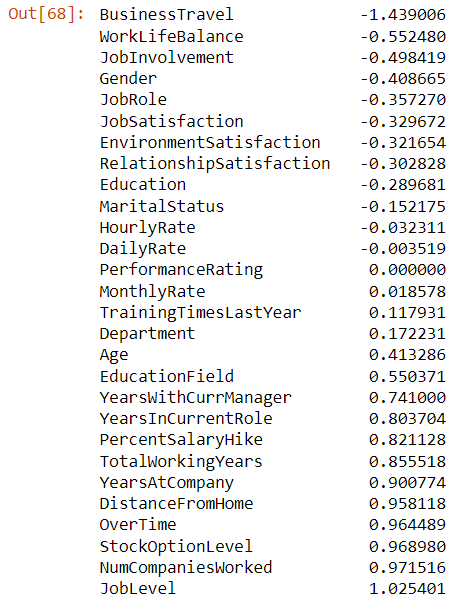
clean\_df['OverTime']=label\_.fit\_transform(clean\_df['OverTime'])

clean\_df['Attrition']=label\_.fit\_transform(clean\_df['Attrition'])

y = clean\_df['Attrition']

x = clean\_df.drop('Attrition', axis=1)

x.skew().sort\_values()



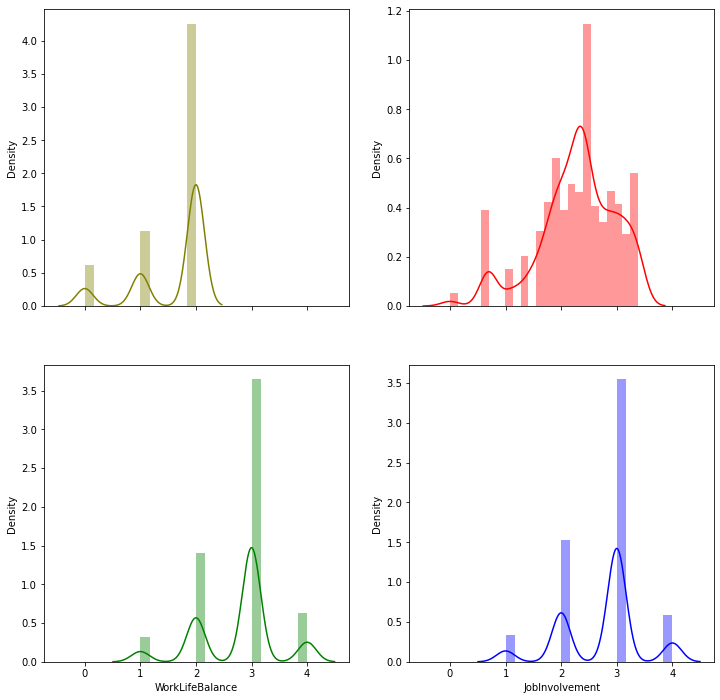
f, axes = plt.subplots(2, 2, figsize=(12, 12), sharex=True)

sns.distplot( x['BusinessTravel'] , color="olive", ax=axes[0, 0])

sns.distplot( x['TotalWorkingYears'] , color="red", ax=axes[0, 1])

sns.distplot( x['WorkLifeBalance'] , color="green", ax=axes[1, 0])

sns.distplot( x['JobInvolvement'] , color="blue", ax=axes[1, 1])



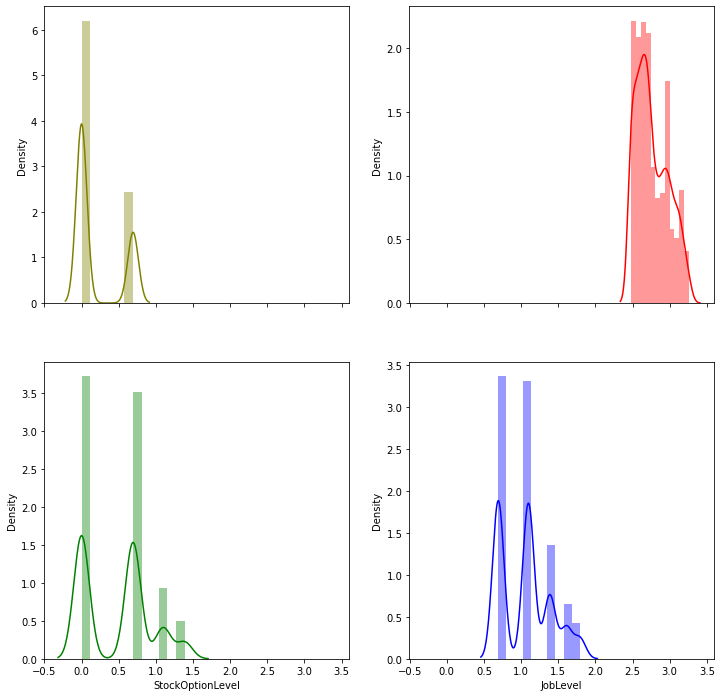
f, axes = plt.subplots(2, 2, figsize=(12, 12), sharex=True)

sns.distplot( x['OverTime'] , color="olive", ax=axes[0, 0])

sns.distplot( x['PercentSalaryHike'] , color="red", ax=axes[0, 1])

sns.distplot( x['StockOptionLevel'] , color="green", ax=axes[1, 0])

sns.distplot( x['JobLevel'] , color="blue", ax=axes[1, 1])



### Feature Selection

As of now, we have dropped some features based on initial observation. Like categorical features with all unique values and ID.

plt.figure(figsize=(20,20), dpi=400)

sns.heatmap(x.corr(), annot=True)

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

def calc\_vif(x1):

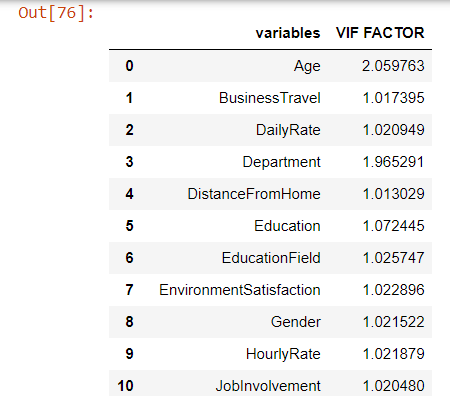
vif=pd.DataFrame()

vif["variables"]=x1.columns

vif["VIF FACTOR"]=[variance\_inflation\_factor(x1.values,i) for i in range(x1.shape[1])]

return(vif)

calc\_vif(x)



**Observation -**

It is observed that, PerformanceRating is having high VIF Factor = 1414.050790 So dropping this column.

x.drop('PerformanceRating', axis=1, inplace=True)

calc\_vif(x)

**Observation -**

It is observed that, Age is having high VIF Factor = 350.509171 So dropping this column.

x.drop('Age', axis=1, inplace=True)

calc\_vif(x)

**Observation -**

It is observed that, MonthlyIncome is having high VIF Factor = 376.093310 So dropping this column.

x.drop('MonthlyIncome', axis=1, inplace=True)

calc\_vif(x)

**Observation -**

It is observed that, PercentSalaryHike is having high VIF Factor = 72.752349 So dropping this column.

x.drop('PercentSalaryHike', axis=1, inplace=True)

calc\_vif(x)

**Observation -**

It is observed that, YearsAtCompany is having high VIF Factor = 47.518006 So dropping this column.

x.drop('YearsAtCompany', axis=1, inplace=True)

calc\_vif(x)

**Observation -**

It is observed that, TotalWorkingYears is having high VIF Factor = 37.070008 So dropping this column.

x.drop('TotalWorkingYears', axis=1, inplace=True)

calc\_vif(x)

**Observation -**

It is observed that, WorkLifeBalance is having high VIF Factor = 14.777019 So dropping this column.

x.drop('WorkLifeBalance', axis=1, inplace=True)

calc\_vif(x)

x.skew()

**Observation -**

Data is some what ready for model building after Standardization

* After checking VIF, decided to drop below features
  + PerformanceRating
  + WorkLifeBalance
  + TotalWorkingYears
  + YearsAtCompany
  + PercentSalaryHike
  + MonthlyIncome
  + Age
* After dropping columns with high multi-collinearity, we can see that there is remarkable drop in skewness of features.
* There are no highly correlated features in columns

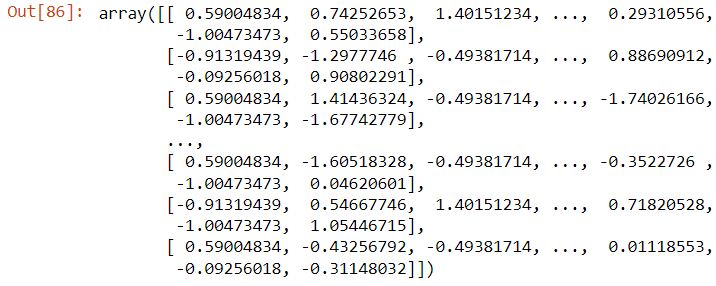
## Building Machine Learning Models

### Scaling

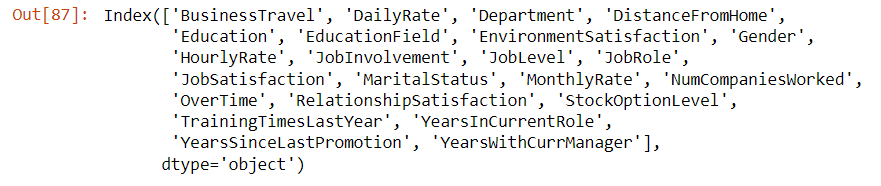
sc=StandardScaler()

x1=sc.fit\_transform(x)

x1

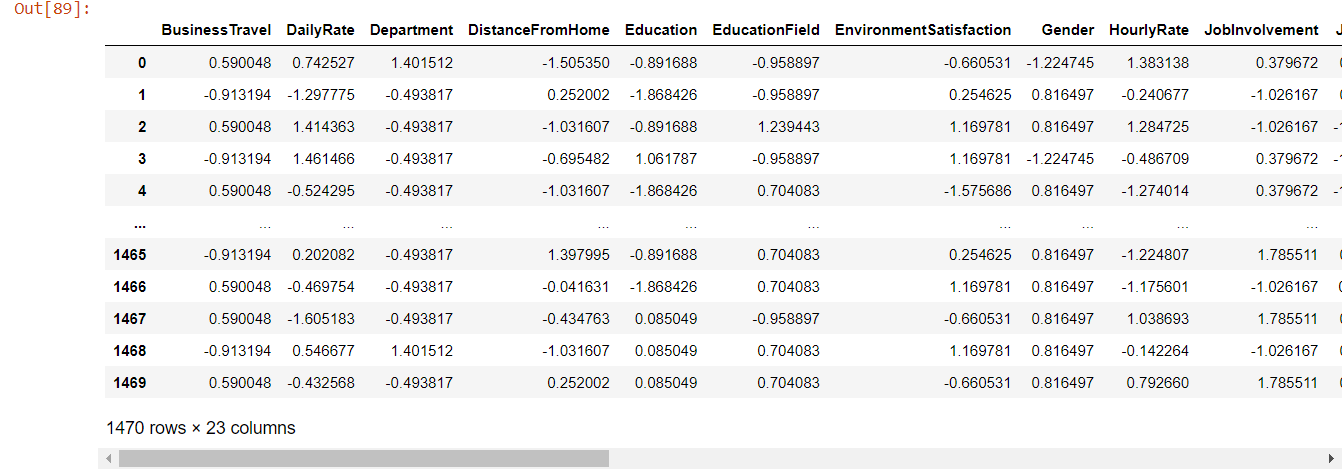


x.columns



x\_scaled = pd.DataFrame(x1,columns=x.columns)

x\_scaled



### Model Building

#### Importing Libraries for Machine Learning

#model developemnt libraries

from sklearn.linear\_model import LogisticRegression

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC, LinearSVC

from sklearn import model\_selection

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import SGDClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import StandardScaler,LabelEncoder

from sklearn.metrics import confusion\_matrix,classification\_report,accuracy\_score,roc\_curve,roc\_auc\_score

from sklearn.model\_selection import train\_test\_split,cross\_val\_score,GridSearchCV

from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier,AdaBoostClassifier

#### Splitting Data Set into Train and Test

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x\_scaled,y,test\_size=0.2,random\_state=43)

#### Model Preparation

dtc=DecisionTreeClassifier()

rfc=RandomForestClassifier()

lg=LogisticRegression()

knc=KNeighborsClassifier()

sgc=SGDClassifier()

seed=7

#prepare models

models=[]

models.append(('DecisionTreeClassifier',dtc))

models.append(('RandomForestClassifier',rfc))

models.append(('KNeighborsClassifier',knc))

models.append(('SGDClassifier',sgc))

models.append(('LogisticRegression',lg))

#### Applying ML Algorithms

#evaluate each model in turn

Model=[]

cvs=[]

score=[]

rocscore=[]

for name,model in models:

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*',name,'\*\*\*\*\*\*\*\*\*\*\*')

print('\n')

Model.append(name)

model.fit(x\_train,y\_train)

print(model)

pred=model.predict(x\_test)

print('\n')

acc=accuracy\_score(y\_test,pred)

print('accuracy score',acc)

score.append(acc\*100)

kfold=model\_selection.KFold(n\_splits=10)

# -----------------------------------------------------------------------

cv=model\_selection.cross\_val\_score(model,x\_train,y\_train,cv=10,scoring='accuracy').mean()

print('Cross-val-score=',cv)

cvs.append(cv\*100)

print('\n')

# -----------------------------------------------------------------------

false\_positive\_rate,true\_positive\_rate,thresholds=roc\_curve(y\_test,pred, pos\_label=2)

roc\_auc=roc\_auc\_score(y\_test,pred)

print('roc\_auc\_score',roc\_auc)

rocscore.append(roc\_auc\*100)

print('\n')

print(classification\_report(y\_test,pred))

print('\n')

cm=confusion\_matrix(y\_test,pred)

print(cm)

print('\n')

plt.figure(figsize=(10,15))

plt.subplot(911)

plt.title(name)

print(sns.heatmap(cm,annot=True))

plt.subplot(912)

plt.title(name)

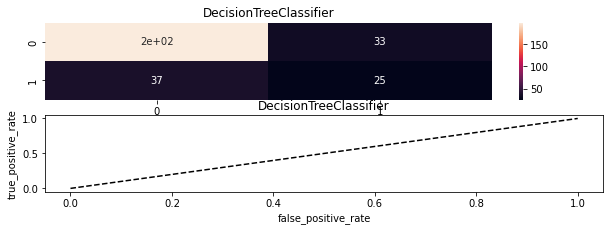
plt.plot(false\_positive\_rate,true\_positive\_rate,label='AUC'%roc\_auc)

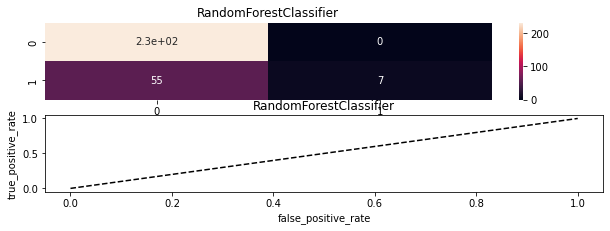
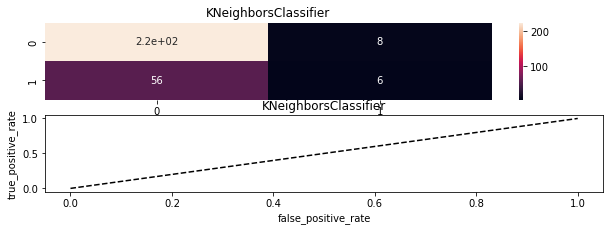
plt.plot([0,1],[0,1],'k--')

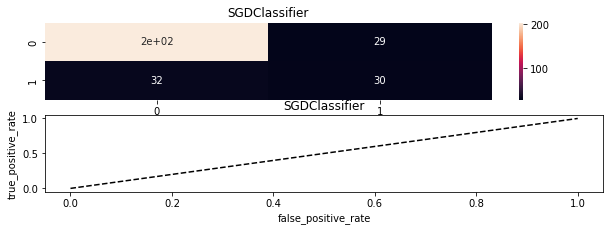
plt.xlabel('false\_positive\_rate')

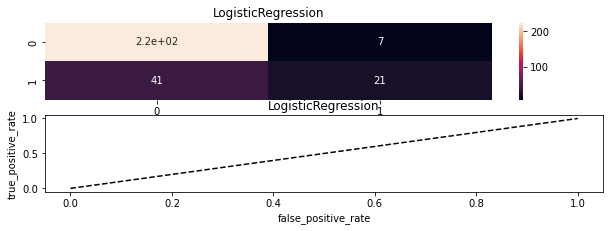
plt.ylabel('true\_positive\_rate')

plt.show()







#### Applying ML Models with Varying random\_State

for i in [0, 1, 101, 47]:

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=i)

# To Prevent Data Leakages, We are scaling data for each train and test. To keep,

sc=StandardScaler()

x\_train=sc.fit\_transform(x\_train)

x\_test=sc.fit\_transform(x\_test)

dtc=DecisionTreeClassifier()

rfc=RandomForestClassifier()

lg=LogisticRegression()

knc=KNeighborsClassifier()

sgc=SGDClassifier()

svc = LinearSVC()

gbc = GradientBoostingClassifier()

gnb = GaussianNB()

seed=7

#prepare models

models=[]

models.append(('DecisionTreeClassifier',dtc))

models.append(('RandomForestClassifier',rfc))

models.append(('KNeighborsClassifier',knc))

models.append(('SGDClassifier',sgc))

models.append(('LogisticRegression',lg))

models.append(('LinearSVC', svc))

models.append(('GradientBoostingClassifier', gbc))

models.append(('GaussianNB', gnb))

#evaluate each model in turn --------------------------------------------

Model=[]

cvs=[]

score=[]

rocscore=[]

for name,model in models:

# print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*',name,'\*\*\*\*\*\*\*\*\*\*\*')

# print('\n')

Model.append(name)

model.fit(x\_train,y\_train)

# print(model)

pred=model.predict(x\_test)

# print('\n')

acc=accuracy\_score(y\_test,pred)

# print('accuracy score',acc)

score.append(acc\*100)

kfold=model\_selection.KFold(n\_splits=10)

# -------------------------------------------------------------------

cv=model\_selection.cross\_val\_score(model,x\_train,y\_train,cv=10,scoring='accuracy').mean()

# print('Cross-val-score=',cv)

cvs.append(cv\*100)

# print('\n')

# -------------------------------------------------------------------

false\_positive\_rate,true\_positive\_rate,thresholds=roc\_curve(y\_test,pred, pos\_label=2)

roc\_auc=roc\_auc\_score(y\_test,pred)

# print('roc\_auc\_score',roc\_auc)

rocscore.append(roc\_auc\*100)

# print('\n')

# print(classification\_report(y\_test,pred))

# print('\n')

cm=confusion\_matrix(y\_test,pred)

# print(cm)

# print('\n')

# plt.figure(figsize=(10,15))

# plt.subplot(911)

# plt.title(name)

# print(sns.heatmap(cm,annot=True))

# plt.subplot(912)

# plt.title(name)

# plt.plot(false\_positive\_rate,true\_positive\_rate,label='AUC'%roc\_auc)

# plt.plot([0,1],[0,1],'k--')

# plt.xlabel('false\_positive\_rate')

# plt.ylabel('true\_positive\_rate')

# plt.show()

result=pd.DataFrame({'Model':Model, 'cvs' :cvs, 'score' :score, 'rocscore' :rocscore}, columns=['Model', 'cvs','score','rocscore'])

print(f'\033[1m ------------------------ Results with Random State {i} ------------------------')

print(result)

sns.barplot(y='Model',x='score',data=result)

result['score']

**------------------------ Results with Random State 0 ------------------**

**Model cvs score rocscore**

**0 DecisionTreeClassifier 77.552513 81.292517 65.102041**

**1 RandomForestClassifier 85.546864 85.374150 56.122449**

**2 KNeighborsClassifier 84.442271 85.034014 56.734694**

**3 SGDClassifier 83.423149 82.653061 64.285714**

**4 LogisticRegression 86.482689 87.755102 68.163265**

**5 LinearSVC 85.885847 87.414966 65.510204**

**6 GradientBoostingClassifier 86.394321 86.054422 63.877551**

**7 GaussianNB 84.698682 85.034014 68.979592**

**------------------------ Results with Random State 1 -----------------**

**Model cvs score rocscore**

**0 DecisionTreeClassifier 80.096335 75.170068 57.225307**

**1 RandomForestClassifier 86.308851 81.632653 56.699299**

**2 KNeighborsClassifier 85.373026 80.612245 54.763296**

**3 SGDClassifier 84.019267 81.972789 61.462595**

**4 LogisticRegression 87.075909 83.333333 59.709234**

**5 LinearSVC 86.396494 82.653061 57.985096**

**6 GradientBoostingClassifier 87.417789 81.292517 60.388662**

**7 GaussianNB 85.803274 82.993197 65.349211**

**------------------------ Results with Random State 101 ---------------**

**Model cvs score rocscore**

**0 DecisionTreeClassifier 77.128785 75.170068 54.390897**

**1 RandomForestClassifier 85.289005 86.054422 55.354752**

**2 KNeighborsClassifier 84.609590 85.034014 54.752343**

**3 SGDClassifier 81.891931 85.374150 62.235609**

**4 LogisticRegression 86.902072 87.074830 66.880857**

**5 LinearSVC 86.563813 86.394558 63.748327**

**6 GradientBoostingClassifier 85.119513 87.074830 65.060241**

**7 GaussianNB 84.100391 86.394558 71.030790**

**------------------------ Results with Random State 47 -----------------**

**Model cvs score rocscore**

**0 DecisionTreeClassifier 77.299725 76.530612 60.413745**

**1 RandomForestClassifier 86.222657 85.374150 55.916900**

**2 KNeighborsClassifier 85.030422 84.353741 55.312062**

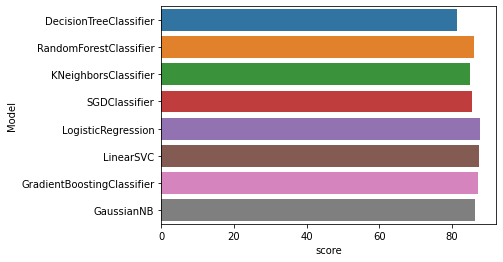
**3 SGDClassifier 83.080545 84.013605 60.422511**

**4 LogisticRegression 86.902072 87.074830 64.007714**

**5 LinearSVC 86.644937 87.074830 63.122370**

**6 GradientBoostingClassifier 86.989715 82.312925 55.873072**

**7 GaussianNB 85.114443 79.251701 61.141304**



### Parameter Tuning

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=1)

clf\_gb=GridSearchCV(estimator=GradientBoostingClassifier(),cv=10,param\_grid=dict({'n\_estimators':[500]}))

clf\_gb.fit(x\_train,y\_train)



clf\_gb.best\_score\_

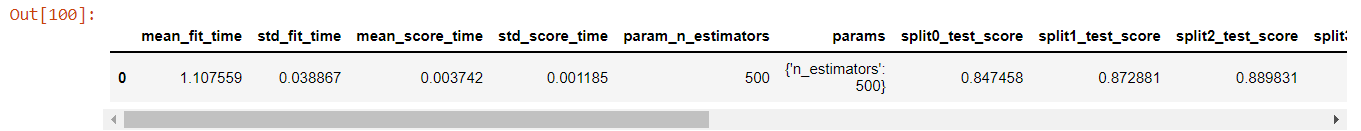


clf\_gb.best\_params\_



clf\_gb\_df = pd.DataFrame(clf\_gb.cv\_results\_)

clf\_gb\_df



pred=clf\_gb.predict(x\_test)

print(f'Accuracy of GradientBoostingClassifier is {round(accuracy\_score(pred,y\_test)\*100,4)}')



### Dmping Model for Production

# Save File

import pickle

file = 'HR\_analytics'

#save file

save = pickle.dump(clf,open(file,'wb'))

## Concluding Remarks

* Employees travelling more are more likely to leave company.
* Among married/Unmarried/Singles – Singles are more likely to leave company.
* Employees from sales department leaves company with higher rate.
* Also employees getting more salary hike are more likely to leave company.
* Freshers with experience of 1-7 years, also tried to switch company.
* Lesser chance of departure - Managers, Research Director, Healthcare Representative
* Higher chance of departure - Research Scientist, Sales Executive, HR, Laboratory Technician
* Doing overtime also plays important role for employee to leave or to continue to work.
* Gradient boosting working good with highest accuracy after parameter tuning.